2014

Low-Power and Programmable Analog Circuitry for Wireless Sensors

Brandon David Rumberg

Low-Power and Programmable Analog Circuitry for Wireless Sensors

by

Brandon David Rumberg

Dissertation submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Electrical Engineering

David W. Graham, Ph.D., Chair
Lawrence A. Hornak, Ph.D.
Vinod K. Kulathumani, Ph.D.
James W. Lewis, Ph.D.
Matthew C. Valenti, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2014

Abstract

Low-Power and Programmable Analog Circuitry for Wireless Sensors

by

Brandon David Rumberg
Doctor of Philosophy in Electrical Engineering

West Virginia University

David W. Graham, Ph.D., Chair

Embedding networks of secure, wirelessly-connected sensors and actuators will help us to conscientiously manage our local and extended environments. One major challenge for this vision is to create networks of wireless sensor devices that provide maximal knowledge of their environment while using only the energy that is available within that environment. In this work, it is argued that the energy constraints in wireless sensor design are best addressed by incorporating analog signal processors. The low power-consumption of an analog signal processor allows persistent monitoring of multiple sensors while the device's analog-to-digital converter, microcontroller, and transceiver are all in sleep mode. This dissertation describes the development of analog signal processing integrated circuits for wireless sensor networks. Specific technology problems that are addressed include reconfigurable processing architectures for low-power sensing applications, as well as the development of reprogrammable biasing for analog circuits.

Dedication

To faculty and peers who have been so helpful,
To family who won't try to read beyond this page—
    and more so to those who will,
And to anyone who may benefit from this work.

Contents

List of Figures

List of Tables

Symbols and Acronyms

ADC	—	Analog-to-Digital Converter
ASP	—	Analog Signal Processor
BPF	—	Bandpass Filter
C^4	—	Capacitively-Coupled Current Conveyor
CAB	—	Computational Analog Block
CB	—	Connection Box
CLB	—	Configurable Logic Block
CMOS	—	Complementary Metal-Oxide Semiconductor
CTS	—	Charge-Transfer Switch
DAC	—	Digital-to-Analog Converter
DSP	—	Digital Signal Processor
FG	—	Floating Gate
FPAA	—	Field-Programmable Analog Array
FPGA	—	Field-Programmable Gate Array
G_m	—	Tranconductance
G_m–C	—	Transconductance-Capacitance
IC	—	Integrated Circuit
κ	—	Subthreshold Slope
LPF	—	Lowpass Filter
LUT	—	Lookup Table
NVM	—	Nonvolatile Memory
PD	—	Peak Detector
PLA	—	Programmable Logic Array
PSRR	—	Power-Supply Rejection Ratio
OTA	—	Operational Transconductance Amplifier
Q	—	Quality Factor
RMS	—	Root-Mean-Square
SB	—	Switch Box
SPI	—	Serial Peripheral Interface
SRAM	—	Static RAM
THD	—	Total Harmonic Distortion
t_{ox}	—	Oxide Thickness
U_T	—	Thermal Voltage
VLSI	—	Very-Large-Scale Integration
WSN	—	Wireless Sensor Network
x_j	—	Source/Drain Junction Depth

Chapter 1

Introduction

The most profound technologies are those that disappear. — Mark Weiser [1]

Advancements in technologies such as MEMS sensors, low-power electronics, wireless communications, and the processing of large data-sets have converged to a point where certain aspects of computing might recede into an invisible ubiquity. By embedding networks of secure, wirelessly-connected sensors and actuators, a conscientious management of our local and extended environments may become second nature. Such a scenario will unlock the capabilities of automation and analytics for non-specialists, and in the process, will help us to make our living and working spaces more comfortable and sustainable, and also help us to make our public and health services more attentive and affordable.

One major challenge for this vision is to create networks of wireless sensor devices that provide maximal knowledge of their environment while using only the energy that is available within that environment. This challenge has prompted a significant amount of low-power circuits research. For example, in the past three regular issues of the Journal of Solid-State Circuits (February, March, and June of 2014), 12 out of 46 papers dealt specifically with this problem of power consumption in wireless sensor devices. Each paper centered upon improving the existing componentry of sensor devices: 5 focused on wireless transceivers, 3 on energy harvesting, 2 on digital circuit techniques, and 2 on analog-to-digital converters.

In this work, we contend that incremental improvements in existing components are insufficient, and that the energy constraints in wireless sensor design are best addressed by incorporating analog signal processors (ASP). The low power-consumption of an ASP allows persistent monitoring of multiple sensors while the analog-to-digital converter, microcontroller, and transceiver are all in sleep mode. By programming the ASP to detect consequential characteristics in the sensor signal, the other components can be awakened as needed. Thus the energy constraints are met without compromising important knowledge of the device's environment.

This dissertation describes the application of analog signal processing to wireless sensor networks and solves several technological problems to minimize the power consumption and to increase the programmability of ASPs. This dissertation is organized into five basic sections:

1. We discuss the backgrounds of analog signal processing and low-power sensing devices (Chapter 2) and then test our "ASP-augmented sensor device" idea with an

automobile-detection application (Chapter 3).

2. We detail the design of two low-power analog signal processing components—a band-pass filter (Chapter 4) and a magnitude detector (Chapter 5).

3. We address one of the major barriers to creating non-trivial analog signal processing systems: programmable analog parameters. To create programmable analog parameters, we use floating-gate transistors, which are the core of Flash memory. To improve the feasibility of floating gates for low-power wireless systems, 1) we have developed a low-overhead floating-gate programming infrastructre (Chapter 6), 2) we have developed better characterization and design-optimization methods for floating gates (Chapter 7), and 3) we have developed an integrated voltage step-up converter to generate floating-gate programming voltages (Chapter 8).

4. We combine much of the above work and revisit our initial test case for an "ASP-augmented sensor device," where we show a great improvement (Chapter 9).

5. We advance this work by developing a field-programmable ASP for low-power sensing applications (Chapter 10) and examine the sensor-network-specific design tradeoffs for field-programmable ASPs (Chapter 11). This reconfigurable architecture allows a larger range of applications and increases the ability to explore new analog signal processing algorithms.

Chapter 2

Analog Signal Processing and Energy Efficiency in Sensor Networks

Large-scale systems of networked sensors offer a real-time understanding of the complex environments that they monitor. Applications include environmental monitoring, protection of borders/resources against intruders, and monitoring of critical infrastructure like bridges and power grids [2–7]. However, wide-scale deployment of sensor networks for these applications has been inhibited primarily by the inability to last for long durations on small power sources, such as batteries and energy-harvesting systems. In this Chapter, we discuss how analog processing can help sensor networks overcome this energy limitation.

2.1 Resource Limitations in Wireless Sensors

Wireless sensor networks are composed of miniaturized computing platforms containing a variety of sensors. The gathered data from all of the sensor nodes are used in combination to provide a detailed understanding of the surrounding environment. Significant variation exists in the hardware design of sensor nodes. While some sensor nodes have been designed for high-bandwidth sensing applications, and have therefore included high-performance processors [8] or FPGAs [9], the majority of research on wireless sensor networks has used low-power "duty-cycle oriented" sensor nodes. These sensor nodes are designed—both in terms of hardware and software—to be active for short durations, and to otherwise occupy a low-power sleep state.

The power consumption of a duty-cycled sensor node includes active power—such as computation, sensing, and transmission—as well as sleep power. The system's total average power consumption can be expressed as

$$P_{\text{total}} = P_{\text{sleep}} + f_{\text{sense}}E_{\text{sense}} + f_{\text{comp}}E_{\text{comp}} + f_{\text{TX/RX}}E_{\text{TX/RX}} \tag{2.1}$$

where P_{sleep} is the sleep power, $f_{\text{sense}}/f_{\text{comp}}/f_{\text{TX/RX}}$ are the frequencies with which sensing/computing/transceiver operations are performed, and $E_{\text{sense}}/E_{\text{comp}}/E_{\text{TX/RX}}$ are the energies used to perform each sensing/computing/transeiver operation.[1] We leave the term

[1]Note that this simple expression does not include the overhead associated with mesh networking. How-

"operation" purposely vague to accomodate a variety of wireless sensing applications. We offer two examples to illustrate different ways to interpret "operation":

1. Consider a temperature-monitoring application in which the temperature is read once per minute. Then, once per hour, the mean, maximum, and minimum of these values are transmitted to a basestation. In this example, one sensing operation consists of measuring and storing one temperature value; one computation operation consists of calculating the mean, maximum, and minimum of 60 values; and one transceiver operation consists of transmitting those three statistics. Furthermore, $f_{\text{sense}} = \frac{1}{60s}$ and $f_{\text{comp}} = f_{\text{TX/RX}} = \frac{1}{3600s}$.

2. Consider a "noise pollution"-monitoring application in which every ten minutes a microphone signal is sampled at 10kHz for one second, after which the A-weighted "loudness" is calculated and stored. Then, all results are transmitted to a basestation once per day. In this example, one sensing operation consists of reading and storing ten-thousand samples, one computation operation consists of calculating and storing the loudness of one 10,000-sample frame, and one transceiver operation consists of transmitting 600 loudness values. Furthermore, $f_{\text{sense}} = f_{\text{comp}} = \frac{1}{600s}$ and $f_{\text{TX/RX}} = \frac{1}{24\times 3600s}$. Note that most low-power sensor nodes have used microcontrollers that are clocked under 10MHz and have included less than 10KB of RAM, so this application would push these systems to their limit while active.

These examples show varying degrees of buffering and compression in each stage of operation. As a result, the complexity and frequency of these operations can be highly decoupled from each other. For example, the frequency of computation events may be much lower than the frequency of sensing events due to buffering, or the frequency of transmission events may be much lower than the frequency of the other events due to compression. Herein lies the application developer's control over power consumption. Once a sensor node platform is chosen, a developer cannot reduce the sleep-mode power consumption (P_{sleep}) or the power consumption in the different operating modes. The total power can only be reduced by reducing the frequency at which the operations occur or by reducing the energy of each operation (e.g. by reducing the complexity, and therefore the active time, of each operation). Consequently, a power-consumption tradeoff exists between the complexity of the operation and the frequency at which that operation occurrs. For instance, in the above noise-pollution example, the relatively high sampling frequency and relatively complex computation operations are balanced by invoking those operations infrequently.

Examining (2.1), we may observe several options for reducing the total power consumption. The frequency of operations (f_*) may be reduced by compressing the data or by taking fewer sensor readings (which may result in an unacceptable loss of data). The sleep power may be reduced by placing components into deeper power-off states; however, the deepest power-off states often do not maintain their operating context and thus require too much time to wake up. The sleep power may also be reduced by optimizing the supply voltage and/or gate threshold voltages to minimize leakage current; however, the leakage current

ever, minimizing the transceiver contributions in the expression will generally help to minimize networking overhead.

should not be made arbitrarily low: an optimal point exists beyond which the energy to perform an operation increases because more time is needed to perform the operation—a rule of thumb says that the leakage power should be approximately 30% of the active power [10].

To reduce the various components of the total average power consumption, sensor network developers have favored platforms that use microcontrollers with low sleep power and fast wake-up from sleep mode. In Table 2.1, we identify three important sensor node platforms. The Mica and TelosB platforms are based upon 8- and 16-bit microcontrollers, and have been used as reference designs for many subsequent platforms, which have similar specifications. Throughout this work, we use the TelosB platform. Some higher-performance 32-bit microcontrollers are approaching the low sleep-power of 8-/16-bit microcontrollers. Ko *et al* [11] have argued that this low sleep-power, combined with greater energy-per-instruction efficiency, allows these higher-performance microcontrollers to achieve lower average power consumption in all but the lowest duty-cycle applications. Consequently, we expect that platforms such as the Opal will become a model for future sensor node platforms.

Table 2.1: Sensor Node Platforms

	Year	MCU	Memory	Wake-up	P_{sleep}	P_{MCU}	P_{Radio}
Mica [12]	2001	ATmega128 (4MHz)	4K RAM, 128K Flash	180μs	75μW	8mW	36mW
TelosB [13]	2004	MSP430 (8MHz)	10K RAM, 48K Flash	6μs	25μW	6mW	60mW
Opal [14]	2011	Cortex-M3 (96MHz)	52K RAM, 256K Flash	10μs	120μW	48mW	94.5mW

For wireless communication, all of the above sensor node platforms have used ZigBee transceivers. The Opal platform uses two ZigBee transceivers in different bands for diversity. ZigBee has been favored for wireless sensor networks because it offers low complexity, low power consumption, and large network sizes. However, these advantages come at the expense of relatively inefficient energy-per-byte compared to other protocols. Currently, many microcontroller manufacturers are combining various wireless connectivity options with Cortex cores (as used in the Opal platform) into single chips. So a greater mixture of wireless protocols may be used in future sensor networks. Table 2.2 summarizes the comparison of wireless protocols from [15]. Regardless of the protocol, the high communication power—compared to the sleep and microcontroller power—compels the use of local compression and decision-making to minimize the total system power.

Table 2.2: Comparison of Wireless Protocols

	Bit rate (Mb/s)	Range (m)	TX (mW)	RX (mW)	Bit rate @ 50μW (kb/s)
ZigBee	0.25	10–100	74	81	0.17
Bluetooth	1	10	103	85	0.49
Wi-Fi	54	10	723	710	3.7
UWB	110	100	750	750	7.3

The power that is available to a sensor node will determine the amount of communication and local computation that it can perform. The power source for most sensor nodes has been a pair of AA batteries. Thoughout this work, we calculate the battery life of our systems by also assuming that the power source is a pair of AA batteries with a nominal capacity

of 1500mAh. However, batteries have temperature and lifetime limitations that prevent sensor nodes from using their full capacity. As a result, much interest has been placed on "supercapacitors"—which achieve several orders of magnitude higher capacitance density than traditional capacitors—as the future energy-storage elements for long-lifetime wireless systems. In contrast to batteries, supercapacitors have lower series resistance, an increased number of charge/discharge cycles, and faster charging capabilities [16].

Regardless of the energy storage element that is used, the energy-storage densities are too low for *small* sensor nodes to endure for decades without recharging. Thus, sensor nodes will need to harvest energy from their environment. As an example, Yerva *et al* [17] showed that, for a node size of 1cm^3 or less, solar cells will supply greater average power than a lithium battery (over a seven-year lifetime, the power would be limited to 10μW). Typical sources of energy that are harvested include photoelectric, thermoelectric, RF, and vibration. Gorlatova *et al* [18] studied the vibration energy that can be harvested from human motion using a 1g proof mass: they found that although walking can generate over 150μW, the average power that is generated over the course of a day by a college student is only 5–10μW. The choice of energy source depends upon the application, but it will likely be common to combine multiple energy sources to guarantee operation in uncertain environmental conditions [19]. While energy harvesting is a promising technology to practically achieve long-life sensor nodes, the power that harvesters supply is insufficient for significant in-network processing.

Returning to the question of power availability for a sensor node: for 10-year operation on a pair of AA batteries, the average power consumption must be less than approximately 51μW, whereas energy-harvesting sources supply in the range of tens to hundreds of microwatts. Let us use 50μW as the target for the average power consumption of the system. To determine the maximum allowable data transmission for a node, Table 2.2 shows the bit rate of different wireless protocols at 50μW power consumption. These numbers are illustrative and assume that the transceivers can operate at less than 0.05% duty cycles with zero sleep power and zero startup power. This is currently unrealistic—e.g. the CC3100 low-power Wi-Fi transceiver requires over 2mW to maintain a connection with an access point. Regardless, we see that a ZigBee transceiver is definitely limited to less than 20 bytes per second. This clearly illustrates the need to compress the data locally for all but the lowest-bandwidth sensing applications.

Turning specifically to the TelosB platform, after 25μW of sleep power is removed from the 50μW budget, only 25μW is left for sensing, computing, and transmission. If this power is split into 8.33μW for each task, then we can read 277sps[2] and transmit 28bps. The remaining 8.33μW for computation must compress 277 samples into 28 bits. This is a challenging task, and we argue in this work that the solution is to use hardware to pre-process the sensor data.

[2]With direct memory access enabled, the TelosB platform can read 200ksps [13]. After duty-cycling to drop the full power consumption to 8.33μW, the sampling rate is 277. We have neglected the power consumption of the transducer that senses the data. In the case of microphones, the lowest-power MEMS microphone (Knowles SPW0430) consumes 240μW. On the other hand, crystal microphones are passive, but they are too large and expensive for the envisioned sensor nodes of the future. Unless otherwise noted, we will assume that a passive transducer can be used, and hope that the power consumption of cheap transducers continues to fall.

Hardware-based event detection has been suggested for reducing power consumption in sensor networks. Referring to (2.1), accurate hardware-based event detection allows all of the "frequency of operation" terms (f_*) to be minimized to the lowest values that allow collection of the important data. The cost of hardware-based event detection is the event detector's power, which is potentially only a minor increase in P_{sleep}. Jevtic *et al* [20] reported a crack monitoring device which uses a comparator to trigger a wake-up signal based on the amplitude of the signal. Their complete wake-up circuit consumes $16.5\mu W$, and they describe the use of both a passive sensor for event detection and a high-precision sensor for event recording. Malinowski *et al* [21] developed a cargo-monitoring tag with a total quiescent current of approximately $5\mu A$. In their event detection circuits, they prepend peak detectors to the comparators, triggering interrupts based on the envelope of the signal. They also describe a dynamically adjustable threshold scheme to achieve a post-event refractory period. Goldberg *et al* [22] presented an acoustic surveillance system, which uses a digital VLSI periodicity detector (with a core power consumption of $835nW$) to wake up the system.

Our approach is to develop a programmable, low-power analog signal processor, which can provide discerning event detection, as well as perform signal analysis to supplement the sensor node's processing capability. Our Netamorph 2.0 analog processor in Chapter 10 can compress and pre-classify sensor data and only adds 20μW of power consumption to a TelosB platform, so that 5μW is left over for communication. This level of power consumption also allows a greater amount of data processing to be achieved with energy harvesting sources.

2.2 Low-Power Analog Signal Processing

It is doubtful that the present course of digital processing will fill the need for local computation in wireless sensors. Power reduction in digital processors has largely relied on device scaling. Device scaling is the regular reduction of transistor dimensions. Beyond the obvious benefit of packing more devices into a given area, Dennard *et al* showed in 1974 that, by proportionally scaling the vertical and lateral dimensions, the power-per-area of digital circuitry would remain constant and thus the computation-per-power would increase with the square of the scale factor [23]. However, this scaling regimen broke down in the late 1990's [24]. Marr *et al* analyzed processor performance-efficiency data from 1980 to 2011 and concluded that the existing trend of exponential improvement is leveling off and will hit a ceiling within the next decade [25]. Nevertheless, there is hope that the overall performance-efficiency will continue to improve for reasons beyond scaling. For a longer-term historical perspective, Koomey *et al* analyzed the overall electrical efficiency of complete computer systems (i.e. including the power supply, monitor, etc.) from the ENIAC in 1946 to personal computers in 2009 [26]. They observed a consistent exponential improvement over a 63-year period—well before the age of CMOS scaling, as well as after the break-down of CMOS scaling. Several disruptive innovations have been required at different times to maintain this trend—transistors, integrated circuits, CMOS logic, switched-mode power supplies, LCD monitors, software-based power management, etc.—and it appears that we have reached the point that a new disruption is needed.

As power constraints on various types of systems are becoming more stringent, analog circuits are being re-investigated for use in low-power systems, such as hearing prostheses

[27] and implantable electronics [28], as well as for the implementation of high-level signal-processing algorithms [29] that are normally performed in the digital domain. Examples of such operations include support vector machines [30], cepstral transforms [31], vector quantizers [32], hidden Markov model decoders [33], motion estimation [34], and adaptive filtering [35]. With such a portfolio of operations, analog circuits can take the place of digital circuits in many signal processing tasks.

Several studies over the past two decades have examined the use of analog circuits to perform low-power processing. Mead made arguments for a computational paradigm that takes advantage of the complexities of the computational primitives (i.e. transistors) [36,37]. Noting the similarities between electron diffusion in subthreshold MOS transistors and ion diffusion in neurons, he made pioneering steps in developing low-power analog circuits that were inspired by biology. Vittoz examined the effects of scaling on the future of analog circuits and analyzed the theoretical performance-per-power limits for linear filter implementations in both digital and analog circuits [38]. He found that analog circuits can often be more efficient for low-precision operations, and he argued that analog circuits are more appropriate for perceptual sensing tasks. Sarpeshkar expanded upon Vittoz's study by considering the cost of general computing operations (i.e. not just linear filters) in analog and digital circuitry, and came to the same conclusion regarding analog circuitry being more efficient for low-resolution operations [39]. Furthermore, his analysis showed that a hybrid analog-digital paradigm—which combines the efficiency at low precision levels of analog circuitry with the bit regeneration and bit-slice scalability of digital circuitry—would achieve the best performance-per-power trade-off. Hasler and Anderson then suggested a "cooperative analog-digital" approach that combines a low-power analog front-end with a programmable digital back-end [29]. They observed that some analog processing circuits at that time (2002) represented a 20-year leap in performance-per-power over digital signal processors. Furthermore, they suggested that the impending data-conversion bottleneck—i.e. the slower performance scaling of analog-to-digital converters (2× improvement every 2 years) in relation to the performance scaling of digital processors (2× improvement every 1.5 years) [40]—could be overcome with a cooperative analog-digital approach, wherein the signal is compressed by the analog front-end in order to reduce the data conversion burden [29].

Despite the advantages of low-power analog signal processing, there have been few advantageous applications of analog signal processing. In the remainder of this work, we apply analog signal processing to wireless sensor networks, and in the process we contribute towards solving the obstacles that are encountered in large-scale, low-power analog signal processing systems.

Chapter 3

A "Hibernets" Event Detector for Slumbering Sensors

Pre-processing of data before transmission is recommended for many sensor network applications to reduce communication and improve energy efficiency. However, constraints on memory, speed, and energy currently limit the processing capabilities within a sensor network. In this Chapter, we describe how ultra-low-power analog circuitry can be integrated with sensor nodes to create energy-efficient sensor networks. To support this proposition, we present a custom analog front-end which performs spectral analysis at a fraction of the power used by a digital counterpart. Furthermore, we show that the front-end can be combined with existing sensor nodes to 1) selectively wake up the node based upon spectral content of the signal, thus increasing battery life without missing interesting events, and to 2) achieve low-power signal analysis using an analog spectral decomposition block, freeing up digital computation resources for higher-level analysis. Experiments in the context of vehicle classification show improved performance for our ASP-interfaced mote over an all-digital implementation.

The work in this Chapter was published in the IEEE Journal on Emerging and Selected Topics in Circuits and Systems [41].

3.1 Introduction

Wireless sensor networks (WSN) hold great promise for use in applications such as environmental monitoring, protection of borders/resources against intruders, and monitoring critical infrastructure like bridges and power grids [2–7]. However, wide-scale deployment of sensor networks for these applications has been inhibited primarily by the inability to last for long durations on small power sources, such as batteries and energy-harvesting systems.

One strategy to conserve energy locally is to perform minimal computation at each node while transmitting most of the data, thereby leaving a majority of the computation and any necessary decision-making to one or more centralized units. However, this strategy leads to increased communication overhead. Therefore, local processing and in-network aggregation are recommended for reducing power consumption due to the high cost of communication. However, the amount of processing that a node can perform is restricted by both its power

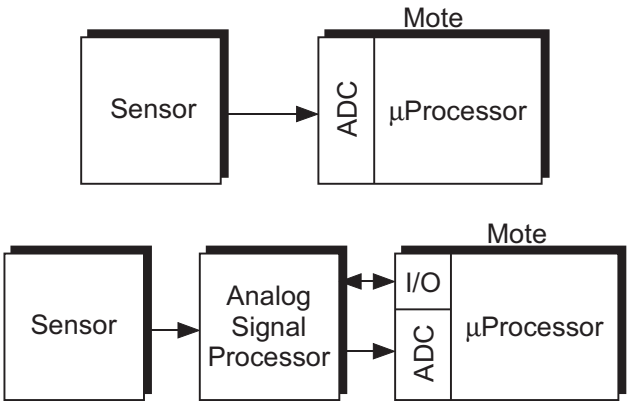

Figure 3.1: In contrast to many WSN designs in which sensor data is directly converted into the digital domain by a mote, we introduce an intermediate stage composed of analog circuits for pre-processing of the sensor data. This analog pre-processor allows us to compress the sensor data into relevant characteristics, improve the performance of event detection (while letting the mote sleep), add processing capabilities, and reduce power consumption.

budget and its limited processing resources. These constraints restrict the amount of signal processing that the node can perform and also limit the highest sampling frequency at which processing can be sustained. In order to perform more advanced signal processing and work with higher frequency signals, it is often necessary for the sensor node, or "mote," to include a faster processor such as in the Intel Imote2 [8] and Stargate platforms, but this technique comes at the expense of higher power consumption and higher cost.

Thus, a fundamental tradeoff exists between the power required to communicate data and the power required to reduce communication using local processing. A variety of network-level design techniques have been developed that trade one for the other in order to increase the life-span of the system [42–58]. While these techniques have yielded useful improvements in life-span, available computational resources limit the degree of those improvements. Significant increases in life-span will require simultaneous consideration of both the hardware and the network-level algorithms.

In this work, we suggest augmenting sensor nodes with an ultra-low-power analog signal processor (ASP). Since analog circuitry offers significant computational resources for minimal power consumption [39, 59], we are using analog signal processing within wireless sensor networks to increase the node-level computational resources while simultaneously reducing the power consumption of these nodes. One of the major objectives of this project has been to develop ways to perform analog pre-processing and classification *prior to* converting into the digital domain [Fig. 3.1 (bottom)], as opposed to immediately converting analog data from a sensor into a digital signal via an analog-to-digital converter (ADC), as is typically done [Fig. 3.1 (top)]. Consequently, we are able to work with the sensor data in its native domain, avoiding unnecessary and power-wasting conversion. Only data that need to be converted are actually processed by the ADC, and only after first being processed/compressed/classified by the analog circuitry.

The outline for this Chapter is as follows. In Section 3.2, we provide a description of our

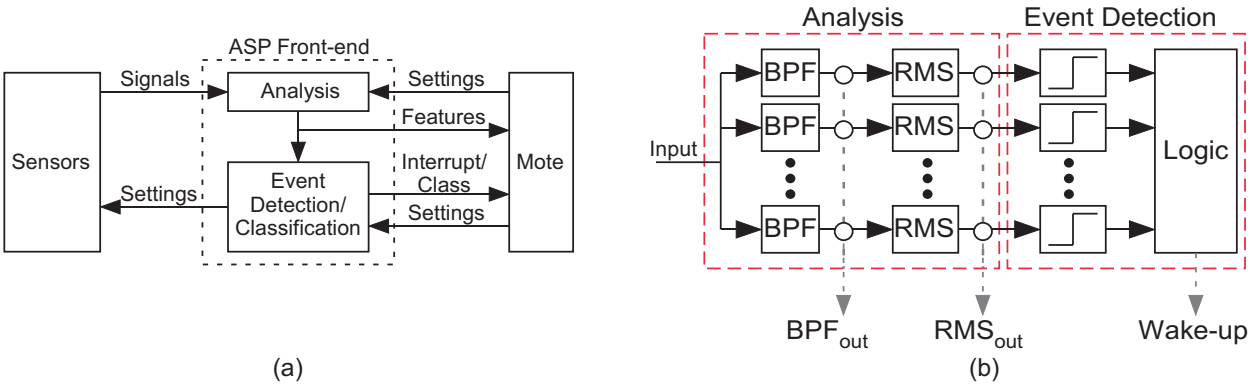

Figure 3.2: (a) A generalized "hibernet" system. In such a system, an analog signal processor (ASP) continuously monitors sensor information while the subsequent digital system (i.e. data converters, microprocessor, and radio) are maintained in a low-power sleep mode. Hence, the higher power consuming portions of the signal-processing chain are allowed to hibernate until needed. (b) The specific analog signal processor that we present in this work is capable of performing spectral decomposition of the incoming signal and generating interrupt signals to wake a sleeping mote.

framework for using ASPs within WSNs to drastically reduce the overall power consumption of a sensor node, and we also provide an overview of how we can apply this technique to acoustic- and vibration-sensing systems. We describe related work in Section 3.3. In Section 3.4, we present our low-power analog circuitry that is used in our ASP, as well as provide demonstrations of analog event detection. In Section 3.5, we discuss how to interface such ASP systems with standard commercially available motes, and then in Section 3.6, we apply this overall system to acoustic classification of automobiles. Finally, in Section 3.7, we discuss our results and summarize our work.

3.2 Analog Signal Processing in Sensor Networks

We present the framework shown in Fig. 3.2(a) as a way to use analog signal processing to simultaneously increase local computational resources while decreasing system-level power consumption. In such a system, the ultra-low-power, "always-on" analog circuitry constantly monitors incoming sensor data to determine if the information is relevant to the system's task. Meanwhile, the digital mote (including the ADC) is kept in a low-power state (e.g. sleep mode). Only when the incoming signal is relevant to the system's task does the ASP trigger the digital system and/or the radio to enter a higher-power state to further process and/or transmit the data.

To perform these wake-up and processing duties, the ASP consists of two parts: 1) signal analysis/pre-processing and 2) event detection/classification. The analysis portion serves two purposes: 1) generate features for use in event detection and 2) perform pre-processing to free up the mote's computing resources. The classifier wakes the mote when it detects events of interest and allows the mote to operate at a higher abstraction level, dealing with

sensor data at the level of classes.

To demonstrate the potential of the ASP/WSN framework, we have designed and fabricated an analog integrated circuit [Fig. 3.2(b)] for use in wireless sensor networks. The analysis portion of the system performs spectral decomposition using a constant relative-bandwidth filter bank with subband root-mean-square (RMS) detection circuits. Event detection is performed using a comparator on the RMS output of each subband, followed by digital logic which asserts a hardware interrupt when the signal spectrum matches a user-defined binary template. The core of this chip operates at an average power of 1–3μW, which is less than the power consumed by a TelosB mote in its lowest-power sleep mode ($> 25\mu$W).

We note that spectral decomposition is a crucial first-step for many sensor network applications, such as acoustic/seismic object classification, event detection, and vibration monitoring [2, 5, 6]. By combining a spectral analyzer with a template-based classifier, our ASP can benefit any application where signal events can be distinguished from other events/noise based on instantaneous frequency content. Therefore, these analog circuits hold great promise for use in wireless sensor networks, and in this work, we show different ways to utilize these circuits.

Typical low power sensor mote platforms, such as the TelosB mote [13], are unable to process incoming data at frequencies higher than a few kilohertz, and are also unable to perform significant signal processing operations such as the FFT [60]. This limitation typically warrants the use of higher-processing-ability platforms, such as the Stargate [61], to perform these signal-processing operations, thus increasing the overall system power requirements. By using the ASP to perform spectral decomposition, we offload major computational tasks away from the mote, thus allowing us to use even a low-power platform such as the TelosB mote. Also, by simply sampling the RMS energy of individual frequency subbands (which can be done at a much lower sampling frequency than the original signal), a mote is able to obtain a complete spectral analysis of the signal. This allows us to operate the system on signals with much higher frequencies than would be possible with a mote alone.

In order for an ASP/WSN system to be practical, the use of the ASP must be as straightforward as writing programming code in a high-level programming language. Also, there should be some flexibility in controlling the parameters of the circuit at run-time and after deployment. With these requirements in mind, we have interfaced the ASP to a TelosB mote [13]. All signal analysis outputs are multiplexed to a single analog-to-digital converter (ADC) pin on the mote, allowing the mote to sample these outputs using standard TinyOS [62] sensor interfaces. The event detector can be set to generate an interrupt when activity has been detected in a user-selected combination of channels. Additionally, the frequency range and spacing of the filter bank can be varied using the mote's built-in digital-to-analog converters (DACs).

We demonstrate the effectiveness of our cooperative analog-digital mote architecture by using it to implement a vehicle classification system similar to [63], and comparing the system performance (accuracy, latency, and energy) with an all-digital implementation. Our chosen application scenario is representative of typical WSN applications for monitoring that involve detection and classification of rare, short-lived events and that demand high accuracy and energy-efficiency. By using an ASP to perform computations and by using the digital mote to refine the classification decisions, we are able to achieve classification accuracies of 90%,

while extending the battery lifespan from four months for a mote-only implementation to nine years for our ASP-mote implementation.

3.3 Background

Many WSN applications require some form of spectral analysis for detection and classification of events [2, 5, 60, 63–65]. All of these applications have discussed the need for processing within the network in order to decrease communication requirements. Our analog front-end would complement all such systems by providing low-power processing capabilities. Additionally, our ASP can complement the low-power digital processors that are being developed for sensor networks [66–69].

Hardware-based event detection, in contrast to sensor polling, has been suggested for reducing power consumption in sensor networks. Jevtic *et al* [20] reported a crack monitoring device which uses a comparator to trigger a wake-up signal based on the amplitude of the signal. Their complete wake-up circuit consumes $16.5\mu W$, and they describe the use of both a passive sensor for event detection and a high-precision sensor for event recording. Malinowski *et al* [21] developed a cargo-monitoring tag with a total quiescent current of approximately $5\mu A$. In their event detection circuits, they prepend peak detectors to the comparators, triggering interrupts based on the envelope of the signal. They also describe a dynamically adjustable threshold scheme to achieve a post-event refractory period. Goldberg *et al* [22] presented an acoustic surveillance system, which uses a digital VLSI periodicity detector (with a core power consumption of $835nW$) to wake up the system. In this Chapter, we present an analog event detector which goes beyond amplitude-based event detection. We also show how the signal analysis performed for event detection can supplement the mote's processing capability.

As power constraints on various types of systems are becoming more stringent, analog circuits are being re-investigated for use in low-power systems, such as hearing prostheses [27], implantable electronics [28], and high-level signal-processing algorithms [29] which are normally implemented in digital, such as support vector machines [30], cepstral transforms [31], vector quantizers [32], bidirectional associative memories [70], and belief propagation [71]. With such a portfolio of operations, analog circuits can take the place of digital circuits in many signal processing tasks, such as acoustic event detection as we discuss in this Chapter. While digital circuits have been used in most settings because of their flexibility, ease of use through programming, noise robustness, benefits of aggressive technology scaling, and scalable dynamic range, analog circuits are able to operate in real-time and perform many computations inherently that would require significant overhead in the digital domain (e.g. multiplication) [59]. Additionally, analog circuitry provides significant power savings over digital, even with the benefits of CMOS scaling for digital systems. For example, it has been observed that ASP performance-per-power represents a 20-year leap over DSP scaling [72], meaning that analog circuitry will continue to provide more efficient signal processing over digital, even though digital processing is progressively becoming more power efficient. This also means that analog has the added benefit of not needing to use the most recent, and often prohibitively expensive, CMOS processes to achieve very low power levels. Instead, analog circuitry can use older and far less-expensive processes and still provide significant

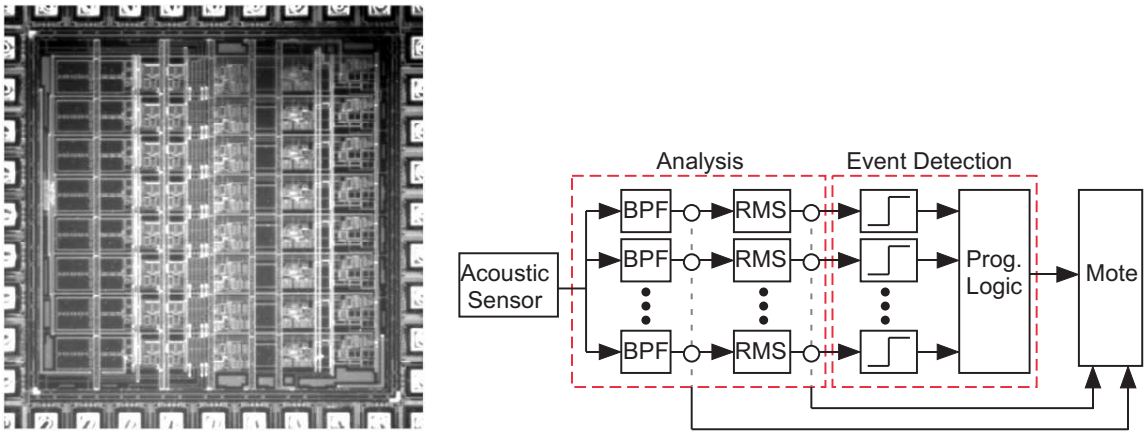

Figure 3.3: (a) Die photograph of the ASP. (b) Block diagram of the ASP. Note that in this preliminary front-end, the programmable logic is external to the IC.

power savings. Furthermore, increased leakage current in smaller processes (i.e. current that flows even when a gate is not switching) requires extra attention to keep power low in newer digital systems and can limit the lowest power state of the digital system [73]; this same leakage current rarely affects continuous-time analog circuits to the same degree since they are typically biased to the exact amount of current required for the application.

While general-purpose microcontrollers dominate most WSN systems due to their flexibility, application-specific (and less flexible) digital circuitry could also be used to perform pre-processing for wake-up tasks (i.e. wake up a more powerful digital system). However, the infrastructure required to support such digital systems can still be quite costly in terms of power consumption. One major advantage of using an ASP as opposed to a digital ASIC for untethered sensing applications is that the sensed signal will inherently be an analog signal. As a result, an ASP can work directly with the signal in its native format. Additionally, a digital system requires data conversion at the full speed of the signal of interest, whereas an ASP approach can reduce/compress the signal content, thereby allowing a further reduction in required power of the ADC. Beyond the necessary ADC, digital systems also require other infrastructure such as a clock, whereas a continuous-time ASP does not, and generating the clock signal will require even further power consumption.

3.4 Design of Analog Computational Elements

Our analog signal processor, shown in Fig. 3.3, is fabricated on a $0.5\mu m$ standard CMOS process available through MOSIS. This integrated circuit is $2.25mm^2$, and consumes only $3\mu W$ when biased for speech frequencies. The intent is to make a low-power, but discriminating, event detector which can call attention to compelling characterisitics of a signal. The detection approach is to identify when the signal matches a binary spectral template. This integrated circuit has two stages: a spectral analysis stage; and an event detection stage, formed by combining an array of comparators with external logic.

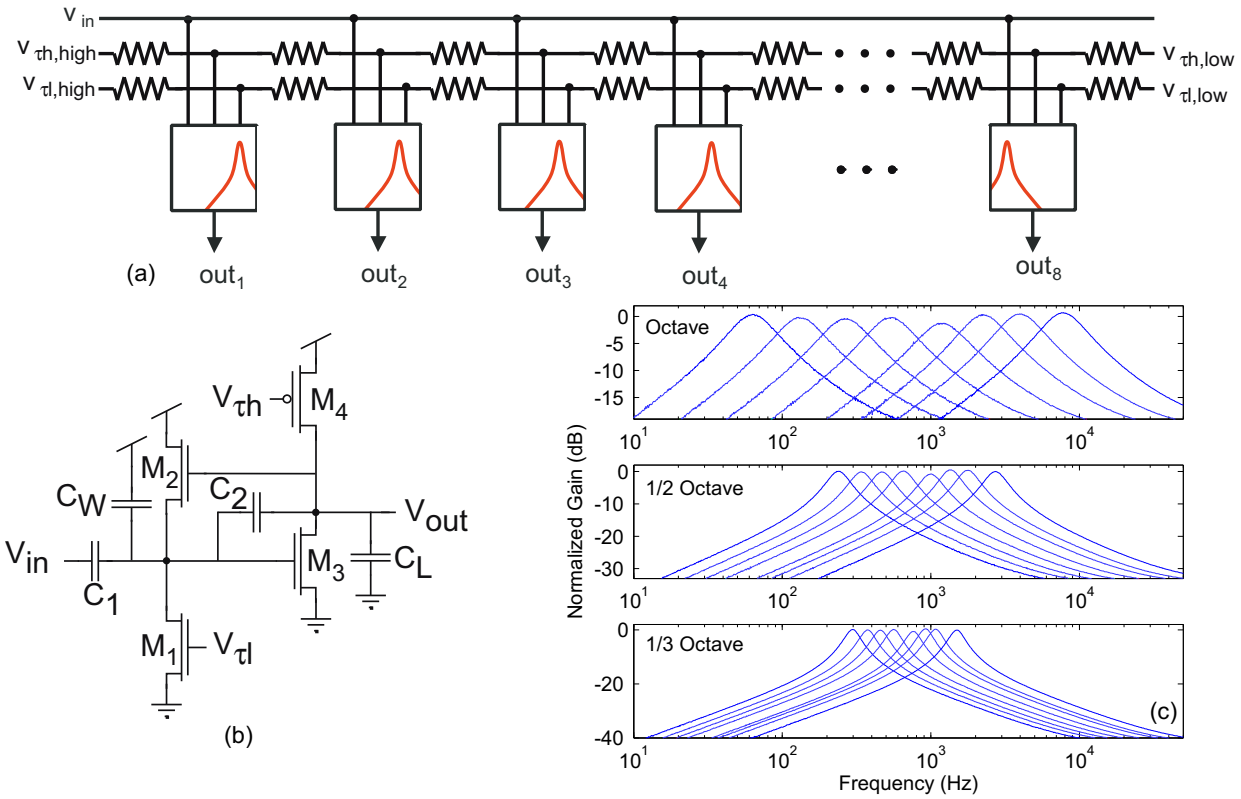

Figure 3.4: (a) Schematic of our filter array and biasing structure. Each of the eight filters receive the input signal in parallel. Two resistive lines are used to bias the corner frequencies of all of the filters. Since the filters are operated in the subthreshold regime, linear spacing of the bias voltages translates into exponentially spaced center frequencies. (b) Schematic of our bandpass filter. The corner frequencies are electronically tunable and are independent of each other; they are established by biasing $V_{\tau l}$ and $V_{\tau h}$, respectively. (c) Frequency response of the filter bank for octave spacing, 1/2 octave spacing, and 1/3 octave spacing.

The spectral decomposition front-end is composed of a filter bank with subband RMS detection circuits. This spectral analysis system is used for frequency-based event detection, and for offloading some of the signal processing which would otherwise be performed by the mote. Since the outputs of all of the filters and RMS circuits are multiplexed to a single pin, a mote can select the filter output or RMS output of any frequency band in order to acquire a frequency-domain representation of the signal.

3.4.1 Filter Bank

The constant-relative-bandwidth filter bank, shown in Fig. 3.4, is created with an eight-channel array of bandpass filters. The filters—schematic shown in Fig. 3.4(b)—are an early version of the filter described in Chapter 4, and so we will forego any details in this Chapter.

3.4.2 Resistive Biasing

Since the filters are operated in weak inversion, the transconductance values of the transistors (and thus the operating frequency of the circuit) vary exponentially with bias voltages $V_{\tau l}$ and $V_{\tau h}$. This exponential relationship between voltage and frequency allows us to achieve the desired log-frequency spacing across the whole filter bank using a simple resistive divider, internal to the chip. The configuration that is used to bias the filter bank is shown in Fig. 3.4(a), where two large resistive lines are used to generate linearly spaced bias voltages for each channel's $V_{\tau l}$ and $V_{\tau h}$, respectively. The voltages on either end of the resistive dividers can be tuned to cover different frequency ranges and spacings, similar to the procedure that was done with early silicon cochlear models (e.g. [74–76]). We use the 1/N octave spacing convention [77], which is common in vibrational and acoustical analyses. In fractional-octave spacing, there are N filters per octave, and the filters cross at their $-3dB$ points. Figure 3.4(c) demonstrates the ability to set the filter bank for one, two, or three filters per octave. These data, and all subsequent data (unless otherwise specified), were obtain from our $0.5\mu m$ standard-CMOS integrated circuit, shown previously in Fig. 3.3.

One significant benefit to using resistive lines for biasing is the ease of use when incorporated into the larger system with the digital mote. In-the-field reconfiguration, which is a highly desirable attribute of WSNs, is easily obtained by connecting the ends of the resistive lines to digitally programmed voltage supplies (e.g. DACs or digital potentiometers). Only a small number of biases must be changed to alter the frequency range and bandwidths of the filters.

While using a resistive divider to bias the filter bank makes the ASP easy to use, there are a few drawbacks. First, the accuracy of the filter parameters depends on the matching of the resistors, which is generally poor. The effects of this mismatch can be observed by looking at the variation in gain across the AC sweeps in Fig. 3.4(c). Second, if using the mote's DAC to permit run-time modification of the biases, resistive biasing will require the mote's DAC to remain turned-on all the time, adding to the quiescent power draw. Both of these issues can be solved by using floating-gate transistors for parameter biasing, as we show in Chapter 4. Floating-gate transistors allow precise programming of each parameter; also, since floating-gate transistors are non-volatile, they do not require any external biasing once they have been programmed. Consequently, in our improved front-end in Chapter 9, we use floating-gate transistors to provide better accuracy and control to our ASP/WSN systems.

3.4.3 Magnitude Detector

For sub-band magnitude detection, we use an early version of the detector presented in Chapter 5. The schematic of this magnitude detector is shown in Fig. 3.5(a). Figure 3.5(c) demonstrates the combination of the filter bank and magnitude detector. In Fig. 3.5(c), our spectral decomposition system is set for 1/2-octave spacing, starting at 250Hz. The input to the filter bank is a logarithmic chirp signal. Shown below the input are the responses of the second, fourth, sixth, and eigth bands of the decomposition system. As the chirp sweeps to higher frequencies, the response of the higher-frequency subbands increases, and the response of the lower-frequency subbands decreases. Note that the output of the magnitude circuit is

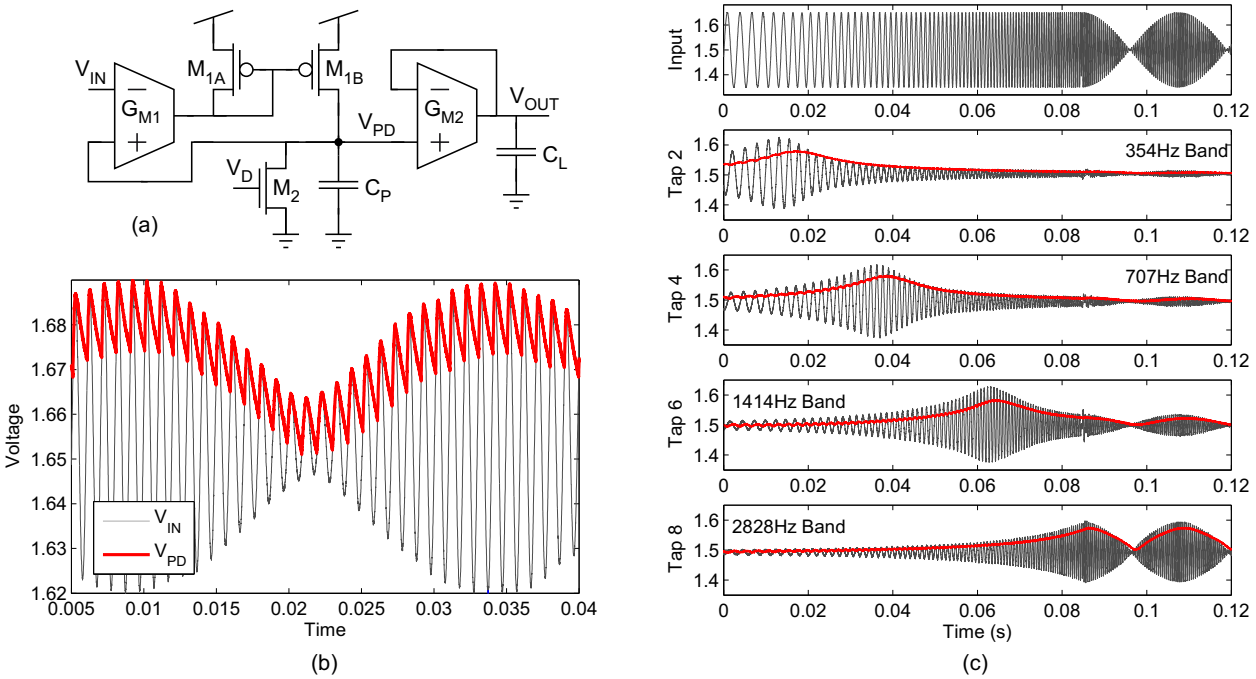

Figure 3.5: (a) Schematic of the circuit used for magnitude detection. (b) Peak detection waveforms. (c) Demonstration of the spectral decomposition front end. The input (top plot) is a logarithmic chirp. The rest of the plots show the response of four of the subbands to obtain an estimate of the RMS of the signal. Note that each channel's response is frequency dependent, and that the RMS outputs represent spectral characteristics of the signal.

the signal content in that band.

3.4.4 Event Detection

By combining the spectral analysis system with comparators and digital logic, we form a simple yet selective event-detection system, with flexibility to define what constitutes an event. Figure 3.6 provides a simple example in which an event is defined as occuring when signal content is present in one of two channels, but not both. The two bands being compared are 500Hz and 1.4kHz. The input consists of a 500Hz sine wave and a 1.4kHz sine wave which overlap for 10 milliseconds. The wakeup signal is generated by combining the comparator outputs for those two bands using an exclusive-or (XOR) operation, so that the interrupt is asserted only when one band exceeds the threshold. In [78], we also illustrated an example in which we detected harmonically related content, which is a scenario that is straightforward to establish using a filter array with $1/N$ octave spacing, such as ours. For example, we defined an event to contain spectral activity in multiple harmonically related bands with the simultaneous absence of spectral activity in non-harmonically related bands.

In these examples, we observe that there is some lag-time between when the event occurs and when the interrupt signal is asserted. The lag-time is caused by the RMS circuit, and is a result of filtering the peak-detected signal. By adjusting the parameters of the RMS circuit,

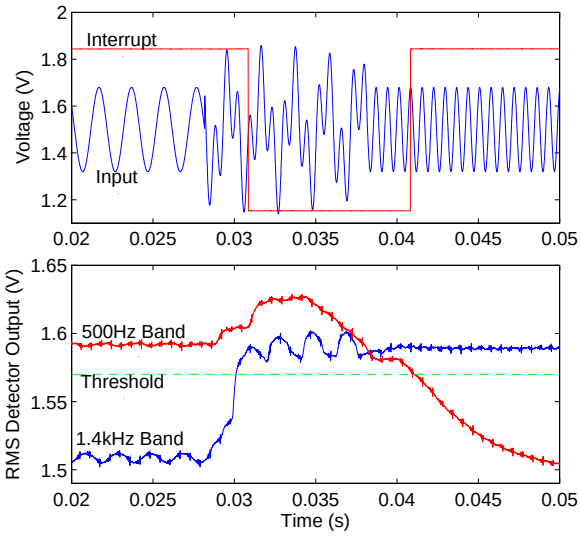

Figure 3.6: Demonstration of multi-band detection using an exclusive-or template. The input consists of two overlapping sine waves. The bottom plot shows the magnitude outputs for the 500Hz and 1.4kHz bands. The comparator outputs of those two bands are combined via an exclusive-or to generate an interrupt when only one band exceeds the threshold.

the phase-lag can be reduced, at the expense of reduced RMS tracking accuracy. This lag time is related to the frequency, f, of the subband, and is approximately $4/f$ for the RMS circuit biasing used in this Chapter. For an application where the mote should record the event, this phase-lag could cause the onset of the event to be overlooked. Regardless of how small the phase lag is, we will miss the prelude to the event. This problem will be present in all systems which wake up based on event detection. To solve the phase-lag problem, the designer can include a memory buffer. This buffer may take the form of an analog delay line (continuous-time continuous-value), an array of sample-and-holds (discrete-time continuous-value), or low-power ADC and RAM (discrete-time discrete-value). This memory can also have a second use of adding memory to the event detection algorithm. Appendix B provides further consideration of such memory buffers.

3.4.5 Power Consumption

The power consumed by our analog integrated circuit is dominated by the bandpass filters, and to a lesser extent, the magnitude circuits. As we presented in [79], which describes the circuit that this BPF is based upon, the power consumed by the BPF is linearly proportional to its center frequency. This relationship is shown in Fig. 3.7 for a filter tuned to a 1/2-octave bandwidth. This relationship enables the system designer to determine the maximum frequency of operation available at a given power budget.

As described for the BPF, the power consumption of the RMS circuit also scales with frequency. Additionally, the RMS circuit can be tuned in various fashions within a given frequency band f_0; for example, this circuit can follow either the envelope or the RMS of a

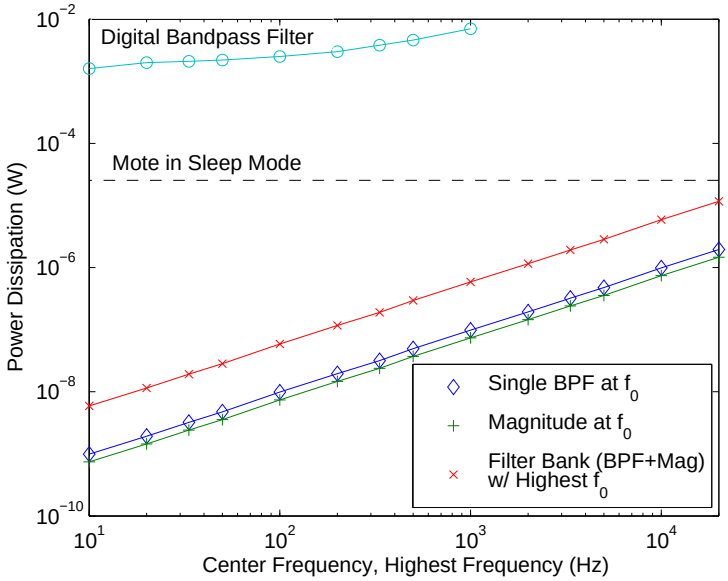

Figure 3.7: The power consumed by our analog spectral-decomposition block depends primarily on the center frequency of the bandpass filter of the highest-frequency subband. The x-axis shows the center frequency for the filter and RMS circuit, and also shows the center frequency of the highest-frequency subband for an array that performs spectral analysis. These are extrapolated from circuit simulations. Also included are power measurements from the digital mote including the minimum measured power consumption in sleep mode and also the power consumption of the mote performing a simple, single bandpass filtering operation (at multiple frequency locations). Note that this mote was unable to simultaneously sample and filter data at frequencies above approximately 1kHz.

signal. Therefore, this circuit has a range of power-consumption values for a given f_0. Figure 3.7 shows the worst-case scenario (i.e. highest power consumption) for operation within a given frequency band, f_0.

The overall power consumption of our analog spectral-decomposition block is set by the center frequency of the highest filter tap. The power consumption of the entire spectral-decomposition system is described by a geometric series, resulting in a total power consumption of

$$P_{tot} = \frac{P_{BPF,high} + P_{RMS,high}}{1 - 2^{-1/N}} \tag{3.1}$$

where $P_{BPF,high}$ and $P_{RMS,high}$ represent the power consumed by the BPF and RMS circuits in the highest-frequency subband, and N indicates the number of filters per octave. The total amount of power consumed by the analog block is shown in Fig. 3.7 for the case of 1/2-octave spacing. Included in Fig. 3.7 is the measured power consumption of the TelosB mote in sleep mode ($25.4\mu W$, which is within the specified bounds of $15 - 60\mu W$). For the entire audio frequency band, our spectral-decomposition block consumes less power than a sleeping mote.

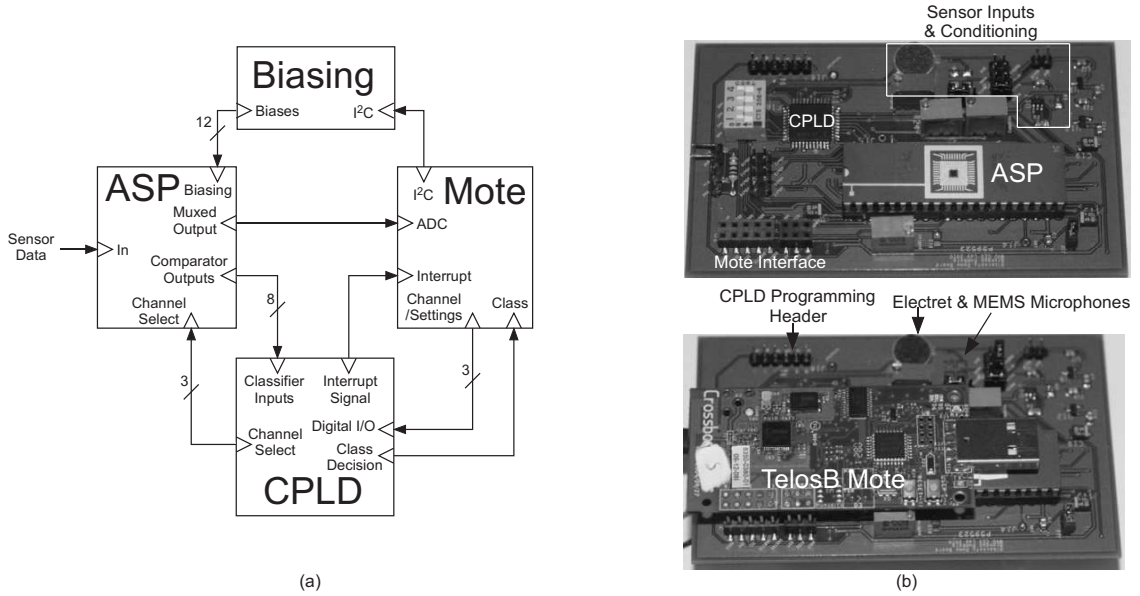

Figure 3.8: (a) Block diagram of the entire system. (b) The complete system shown with and without the mote connected to it.

3.5 Interfacing With the Telos Mote

To evaluate the potential of the ASP/WSN framework, we interfaced the integrated circuit described in Section 3.4 with the TelosB mote. The TelosB mote was chosen for its low-power sleep mode (25 μW) and fast wakeup time (6 μs), which make it suitable for a hardware-based wake-up system. A printed circuit board [Fig. 3.8(b)] was built to combine all components of the system [Fig. 3.8(a)]. A summary of the power consumption of the circuit board is shown in Table 3.1.

Two acoustic sensors are included on the circuit board, both an electret microphone and a MEMS microphone. Additionally there is an auxiliary interface for connecting different sensors, such as the passive piezoelectric microphone which we use as the main microphone. Included is a low-power microphone amplifier based around the MCP6141 operational amplifier. The sensor output is available as an input to both the ASP and the mote's ADC.

As mentioned in Section 3.4, the filter bank and subband processing elements are biased with a resistive divider. On the circuit board, the voltages at the ends of these resistive lines are provided by a network of digital potentiometers (AD5263) and two low-power voltage references (the ISL60002 and the REF3318). In addition, the comparator trigger point is also set via the digital potentiometers. A resolution of approximately 2.5 mV is available across the nominal range of bias voltages for each circuit. The mote applies new settings to the potentiometers after receiving updates over the radio.

A complex programmable logic device (CPLD) was used for the hardware-based pattern classifier. The XC2C32A was chosen due to its low power consumption. The CPLD arbitrates all digital connections between the Mote and ASP, and serves two roles: 1) it implements the detection rules that operate on the comparator outputs, i.e. it performs the role of the "logic"

in the event detector [Fig. 3.2(b)], and 2) it serves as a serial-to-parallel converter, allowing us to use just three of the mote's digital I/O pins to select which of the ASP's analog outputs are connected to the mote's ADC, and also to choose between different sets of detection rules which are preloaded into the CPLD. For event detection, the CPLD receives the comparator outputs from all frequency bands, and performs template matching to detect/classify events. Upon detecting an event, one of the CPLD output pins wakes up the mote via a hardware interrupt, and the other output pins indicate the classification of the event.

To provide the mote with access to the ASP's signal analysis, the outputs from all bandpass filters and RMS circuits are multiplexed to the mote's ADC. The mote communicates through the CPLD to specify which subband is connected to the ADC. To acquire the entire spectral representation, the mote selects a new subband between each sample. Due to the low-frequency nature of the RMS outputs, the mote is able to cycle through all channels without experiencing aliasing.

3.6 Performance Evaluation

In this section, we describe two modes in which we can exploit the computational capabilities of the analog integrated circuit for WSN applications, namely 1) selective wake-up mode and 2) selective sample mode. We then quantify the performance gained in both cases. Finally, we demonstrate the use of our ASP-interfaced mote in a vehicle classification application and highlight the energy efficiency gained in comparison to an all-digital implementation.

3.6.1 Selective Wake-Up Mode

In the selective wake-up mode, we take advantage of the low-power processing capability of analog circuits by placing the mote into long periods of hibernation and then selectively waking the mote when a user-specified combination of frequency components are present in the signal. Figure 3.9(a) demonstrates single-band event detection. The band of interest is 1kHz and the filter has a quality factor of 2.8. Signal content appears in the band at 2.6 seconds but has noise added to it. This noise is a combination of white noise and tones at 100Hz, 600Hz, and 10kHz. The bandpass filter focuses on the frequency of interest and the comparator trips once the RMS reaches the threshold. Note that the subband event is detected despite having much lower amplitude than the noise and other frequency components. In this mode, the mote samples the raw sensor signal when it wakes up and transmits it to a basestation. The signal received by the basestation is shown in the bottom trace.

In order to compare the power consumed by the ASP-interfaced mote with a mote-only implementation, we implement a second-order Butterworth bandpass filter on a TelosB mote running TinyOS and measure the power consumed. The measurements are taken with a stock TelosB mote, without any of the components added for ASP-interfacing. The digital filter is implemented by buffering 100 samples at a time and then computing the filter outputs after every 100ms. The power consumed for this operation is measured for sampling frequencies ranging from 10Hz to 1kHz. In Fig. 3.7, we compare the average power consumed

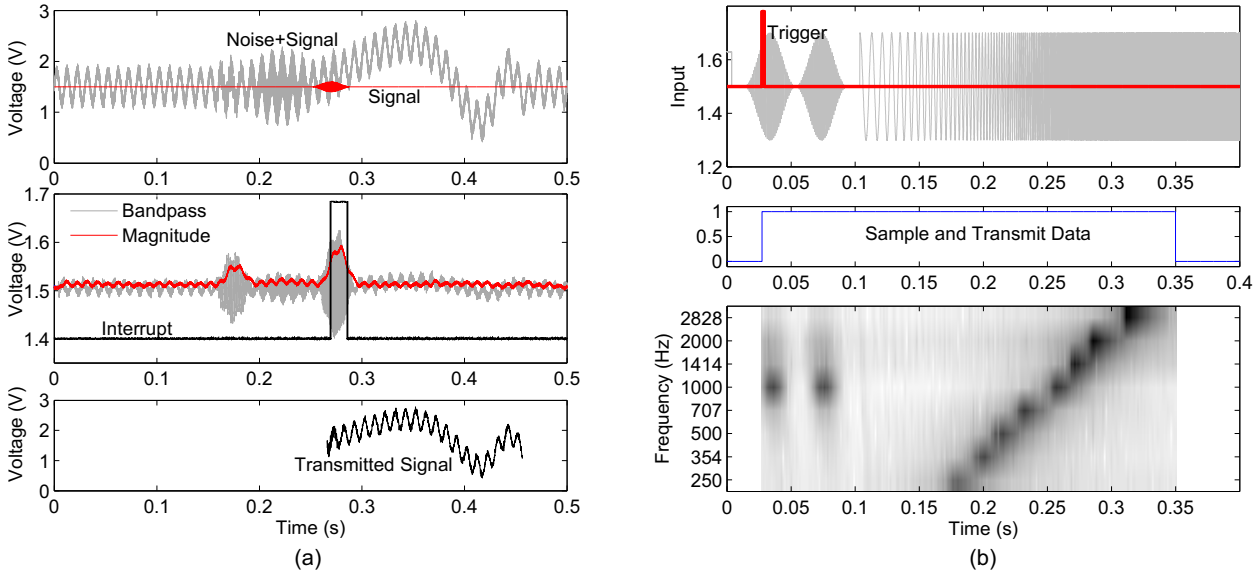

Figure 3.9: (a) Single-band event detection. The frequency of interest is 1kHz and is the "Signal" trace in the top plot. Broadband noise and tones at 100Hz, 600Hz, and 10kHz are added to "Signal," generating "Noise+Signal," which is the input to the analog signal processor. The middle plot shows the response of the three stages of the processor within the 1kHz subband. The bandpass filter cuts the undesired frequencies, while the RMS circuit tracks the magnitude. Once the magnitude exceeds the threshold, an interrupt is generated to wake the mote. The mote then samples the output of the sensor and transmits it to the basestation. The received signal is plotted in the bottom plot. (b) Sampling of pre-processed subbands and spectral analysis performed by the analog IC. (*Top*) The input signal is composed of two 1kHz pulses followed by a logarithmic chirp signal. (*Middle*) Once the trigger goes high, the ADC samples all 8 channels for a user-specified amount of time (e.g. 300ms). (*Bottom*) Spectrogram of the transmitted frequency-dependent magnitude data as received by the base station.

by the digital filter with the power drawn by an analog bandpass filter for different sampling frequencies and center frequencies, respectively. No data points could be obtained for mote power at frequencies above 1kHz since that is the highest sampling frequency that the TelosB can simultaneously sample and filter data. We point out that the energy consumed by our entire spectral analysis system is over 1000 times lower than the power consumed by a single digital filter, thus signifying the energy savings compared to keeping a mote always turned on.

3.6.2 Selective Sample Mode

In the selective wake-up mode described in the previous subsection, once the mote is awake it samples the raw signal for processing or transmission. The drawback with this approach is that a low-power processing platform such as the TelosB mote is unable to sample and process signals of high frequencies and is also limited in the kind of signal

processing operations that can be performed (an FFT, for example, is infeasible on a TelosB mote [60]). This often warrants the use of a platform with greater processing capabilities, such as the IMote2 or Stargate [61], for performing these signal processing operations, which increases the overall power requirements of the system. In this subsection, we highlight the selective-sample mode of operation in which we take advantage of the ASP's ability to perform pre-ADC signal analysis. The ASP is used to perform a full spectral analysis of the input signal and the mote only samples the RMS energy of each subband. Thus we are able to reduce the computational resources required at the mote, allowing for lower power operation.

In the experiment of Fig. 3.9(b), the input signal consists of two 1kHz pulses followed by a chirp signal. The 1kHz pulse is used to trigger the mote into sampling the RMS energy of each subband in succession, for a specified period of time. The mote scans through subbands by writing to the GIO port's output register between each sampling operation. The frequency-decomposed RMS data obtained by the mote is transmitted to a base station and is displayed in the bottom plot. We note that by scanning through the energy of all the subband channels in succession, a complete spectral decomposition can be obtained at the mote in real time using the analog circuit. By doing so, we are also able to operate the system on signals with much higher frequencies (since we sample only the RMS amplitude of sub-bands) than would be possible with a mote alone.

3.6.3 Evaluation in the context of an automobile classification application

In order to evaluate the accuracy and energy-efficiency of our ASP-interfaced mote in the context of an actual sensor network application, we have used the system in a re-creation of the all-digital acoustic-sensor-based vehicle classification experiment that we described in [63]. The vehicle classification system is intended for unattended monitoring of secure facilities. The objective of the system is to accurately identify an approaching vehicle as belonging to one of multiple categories, such as small, medium, and large vehicles, and then accurately raise an alert when a vehicle of a particular type has been detected. The vehicles are assumed to appear in isolation and not concurrently with other vehicles. The system is required to have a long lifespan on battery sources, while at the same time retaining high accuracy and low latency in classification. Arrival of any vehicle is expected to be a rare event, therefore rendering duty cycling of resources essential for energy-efficiency – but at the same time it is critical that no vehicles are missed. We note that the chosen application is representative of typical wireless sensor network applications for monitoring, such as detection of anomalies in bridges [80], unattended ground sensing by military personnel in combat situations, classification of objects for asset protection [2], classification of animal sounds [81], and monitoring of seismic activity. All of these applications involve detection and classification of rare, short-lived events and demand high accuracy and high energy-efficiency.

In this subsection, we describe the implementation of the vehicle classification system described above using our ASP-mote architecture and compare the system performance of our cooperative analog-digital implementation with that of an all-digital implementation. We specifically consider classification into two vehicle categories: car and truck.

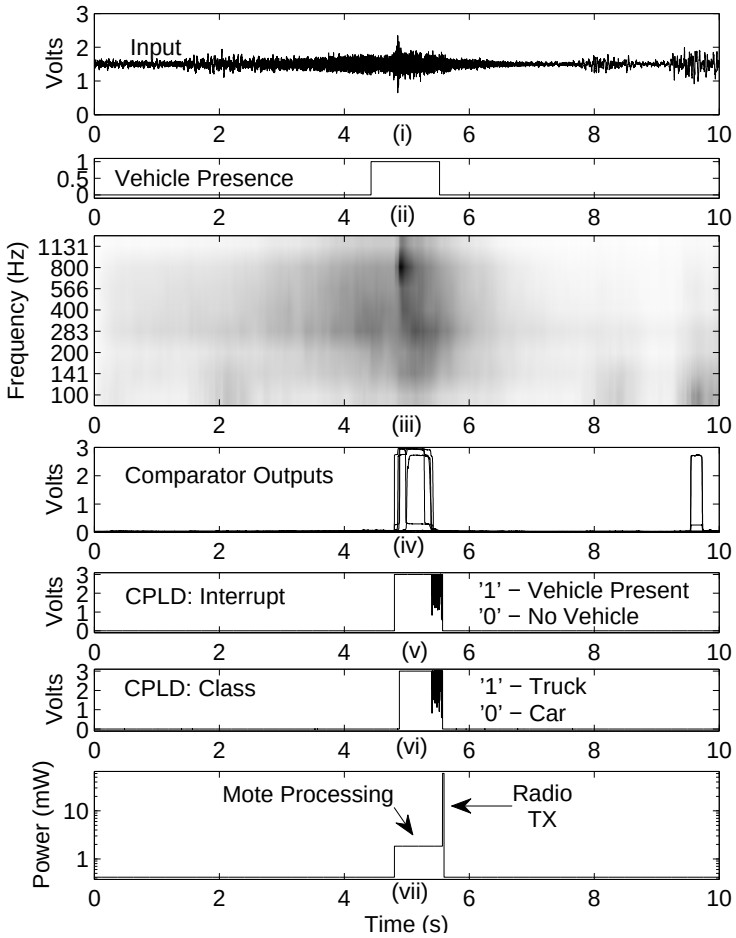

Figure 3.10: Demonstration of the stages of the detection system for a 10 second test sample of a truck being classified. The truck is closest to the sensor between seconds 4–6 of the test sample [shown in (ii)]. The comparator outputs of the 8 filter-bands are shown in (iv) and the CPLD outputs are shown in (v) and (vi). The CPLD interrupt pin goes high when a car or truck is detected. The CPLD class pin specifies the classification (high for truck, low for car) and is only valid when the interrupt pin is high. Once the interrupt is generated, the mote is awakened and starts recording and accumulating the CPLD classifications (consuming about 1.5mW of power). When a final decision is made, the output is transmitted via radio, which consumes 60mW [shown in (vii)].

3.6.3.1 Data Collection

The dataset collected for the experiments described in [63] was used for performance evaluation in this work. The acoustic sensor used for data collection was a Samsung C01U - USB Studio Condenser Microphone. The directional microphone was placed 10-12 feet from the road, mounted one foot off the ground, and combined with Samson windshields to filter out wind noise. A mid-sized car and a pickup truck were considered as the two vehicle classes. Multiple observations were collected for both vehicles, which were driven at speeds

between 10mph and 30mph. Ambient data was also collected using the microphone without any vehicle being present in the scene.

3.6.3.2 Training

The dataset was first normalized so that the peak amplitude of the signal across vehicle classes was uniform. The dataset was then divided into two sets, one for training and the other for testing, and regions of the data corresponding to when the vehicle was and was not present were manually identified. Based on the short-time FFT spectra of the data, the ASP's filter bank parameters were chosen to be half-octave spacing from 100Hz to 1131Hz. Using these filter bank settings, analysis was performed on all of the training samples by streaming them through the ASP using a DAC and recording the RMS output of each subband. After obtaining the RMS data, the objective was to determine the combination of comparator trigger point and codeword assignments (codeword defined as the 8-bit output from the eight comparators) which achieves the desired classification performance. During training, each of the possible 256 codewords were associated with a class (i.e. car, truck, and no vehicle).

The training procedure, which was performed offline, was to iterate through comparator threshold values (20 steps of 10mV), performing the following steps for each threshold: 1) thresholding was applied to the RMS data to obtain an 8-bit codeword for each time step, 2) the distribution of each class (i.e. car, truck, and no vehicle; combined across all observations of the class) across all codewords was computed, 3) each codeword was assigned to the class that was most likely to result in observing that codeword (i.e. the class that caused that codeword for the largest percentage of time), and then 4) the combination of comparator threshold and codewords was evaluated by finding the percentage of time-samples which were associated with the correct class. After iterating through the threshold values, the threshold which resulted in the largest percentage of correct decisions in step 4 was chosen as the final threshold and the codeword assignments found in step 3 for that threshold were used as the final codeword assignments.

Once the comparator threshold and codeword assignments were determined, the system was configured by transmitting the threshold value to the mote and programming the codeword assignments into the CPLD via the JTAG header on the circuit board. The CPLD was programmed such that the interrupt pin went high whenever a codeword associated with either a car or truck was encountered, and the classification pin went high whenever a truck was encountered.

Note that the instantaneous categorization generated by the ASP is susceptible to false decisions due to noise or differences in the "approaching" versus "present" sounds of the vehicle. Hence, it is possible for an interrupt pin to be reset despite the presence of a vehicle, causing the GPIO pins to provide a false classification. In order to compensate for these false decisions, we use the mote to generate the final classification output based on inputs from the ASP over a length of time. Once an interrupt has been generated by the ASP, the mote stays on and records the state of the interrupt pin and the GPIO pin until the interrupt stays low continuously for a duration of 100ms, confirming that the vehicle is outside of the sensing range. The mote then generates the final classification result as the most frequent decision from the ASP over the duration of the event. This simple decision-accumulation

scheme provides good classification results; however, the scheme increases latency since it waits until the vehicle has left the sensing range before making a decision. Alternative schemes may be used to make the decision sooner, and future versions of the ASP will include decision-accumulation capabilities to avoid waking the mote prematurely.

3.6.3.3 Testing

All testing was performed by streaming the samples into the ASP using a 16-bit DAC at a samping frequency of 4kHz. The operation and the power consumption of the ASP-interfaced mote is shown in Fig. 3.10 in the form of a timing diagram for one 10-second test sample of a truck [Fig. 3.10(i)] being classified. The truck is closest to the sensor between seconds $4-6$ of the test sample [shown in Fig. 3.10(ii)]. The spectral analysis output of the event detector front-end is shown in Fig. 3.10(iii) in the form of a spectrogram. The comparator outputs of the eight filter-bands are shown in Fig. 3.10(iv), and the CPLD outputs are shown in Fig. 3.10(v)-(vi). The CPLD interrupt pin goes high when either a car or truck is detected, and the CPLD class pin specifies the classification (high for truck, low for car), which is only valid when the interrupt pin is high. Once the interrupt is generated, the mote is awake and starts accumulating the classifications from the CPLD (consuming about 1.5 mW of power). When a final decision is made, the output is transmitted via radio (if it was determined that an event occurred), which consumes 60mW [Fig. 3.10(vii)]. The detailed power consumption of the ASP-interfaced mote for the various operations being performed are shown in Table 3.1. The accuracy of classification is highlighted in Table 3.2. An overall accuracy of 90% is achieved with an average false alarm rate of one false positive every 50 seconds in the presence of amplified ambient wind noise.

3.6.3.4 Comparison with all-digital implementation

Low-power computing platforms such as the TelosB mote are unable to perform spectral analysis on-board, and therefore processing platforms such as the Stargate have to be used to perform signal processing. Since these devices consume significantly higher power, they are typically used in a layered architecture in conjunction with mote platforms that act as wakeup devices to trigger the detection of an event. In [63], we presented an all-digital implementation using such a layered architecture for the vehicle classification system described above. In that all-digital implementaion, a low-power Mica2 mote attached to a seismic sensor stays on all the time to detect the arrival of a vehicle. Upon detection of a vehicle, the mote triggers a signal to wake up a Linux-based Stargate platform that performs spectral analysis for vehicle classification. The Mica2 mote stays on all the time and consumes 24mW of power when processing and 60mW when transmitting. The Stargate running off of a 4.2V battery consumes 420-470mA when processing for a duration of $8-10$ seconds per vehicle detection. In comparison, our cooperative analog-digital implementation consumes only 214 μW of power when idle and 1.5 mW when an event is detected.

Now we analyze the power savings afforded by using the ASP in the vehicle classification scenario. Table 3.1 details the contribution of each component to the system's power budget, showing the power breakdown of the ASP-augmented mote for three operating states: event-monitoring (mote asleep while ASP performs event detection), sampling/processing

Table 3.1: Power Consumption

	Device	Power	Power (Projected)
ASP-mote event-monitoring	ASP	3μW	3μW
	CPLD	48μW	5μW
	Biasing	135μW	20μW
	Sensor (w/ interfacing)	3μW	3μW
	Sleeping mote	25μW	25μW
	Total	214μW	56μW
ASP-mote sampling/ processing	ASP board	189μW	31μW
	I/O Buffers	30μW	30μW
	Awake mote	1.5mW	1.5mW
	Total	1.72mW	1.56mW
Transmitting	ASP board	189μW	31μW
	Transmitting mote	60mW	60mW
	Total	60.19mW	60.03mW

Table 3.2: Vehicle Classification Results

	Ground Truth		
	NULL (200 seconds)	Car (10 Samples)	Truck (10 Samples)
NULL		20%	0%
Car	2 false alarms	80%	0%
Truck	2 false alarms	0%	100%

(mote awake, e.g., the decision-accumulation scheme discussed in Section 3.6.3.2), and data transmission. Since this initial IC did not have optimized biasing and did not integrate the logic that is needed for the pattern matching portion of the event detector, we provide two sets of power numbers: the measured values for this system which are shown under "Power," and the expected power values (assuming integrated event-detection logic and floating-gate biasing) which are shown under "Power (Projected)." The projected power numbers are based on previous floating-gate-biased filter banks and programmable logic arrays that we have made. The "Sensor (w/ interfacing)" number is for a passive piezoelectric microphone with a low-power amplifier.

To visualize the power savings of the event detector, we plot the system lifespan as a function of the frequency of events (Fig. 3.11), assuming a nominal battery capacity of 1500mAh. Using the power numbers given in Table 3.1, the lifespan is calculated for the following platforms: an ASP-augmented mote (w/ measured power numbers), an ASP-augmented mote

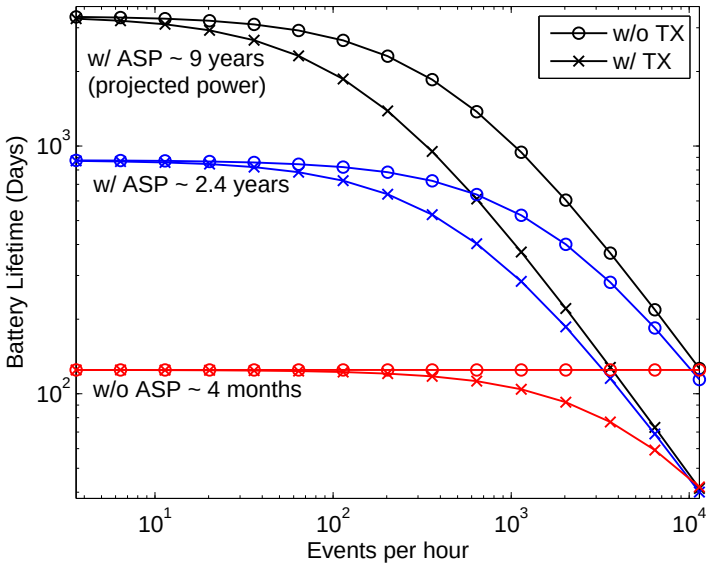

Figure 3.11: System lifetime as a function of event frequency.

(w/ projected power numbers), and a digital-only mote. For our comparisons, the digital-only mote is the TelosB, which is one of the lowest-power commercially-available mote platforms. Comparing against the TelosB mote gives digital-only platforms the benefit of the doubt since (as we discussed in Section 3.6.1) the TelosB is unable to perform even a single band-pass filter in real-time for signal bandwidths exceeding 1kHz. Each platform is considered with and without the cost of transmitting the classification decision (which requires the radio to be turned on for 16ms). The digital-only scenario with no transmission has a constant lifetime since it is always awake and processing, while the digital-only scenario with transmission shows a decreasing lifetime with increasing event frequency since the radio is turned on more frequently. In the ASP scenario, the mote enters a low-power state between events. When events are infrequent, the average power consumption of the system approaches the sum of the ASP and sleeping-mote power levels. As events become more frequent, the mote spends a larger percentage of time awake, and the system's average power consumption approaches the sum of the ASP and awake-mote power levels. When events occur so rapidly that the mote never turns off, the lifespan of the ASP-augmented system drops slighly below the lifespan of the mote-only system due to the additional power of the ASP.

3.6.3.5 Discussion

We note that in our implementation, we used the ASP to output binary decision bits, which are read with the GPIO pins on the mote and used to make the final classification output. Alternatively, the mote can be used to sample the RMS energy of each subband, as described in Section 3.6.2, while the interrupt pin stays high, and then use the sampled spectrogram of the signal to make a decision. Such an approach is likely to be beneficial in a more general classification scenario with much more than 2 classes.

3.6.4 Other Applications and Potential Extensions

We chose to implement spectral decomposition as our computational block in this version of our ASP because spectral analysis is often the first step in a majority of WSN event detection/classification applications (e.g. vibration monitoring for anomaly detection in buildings, bridges, etc. [6], vehicle classification [5,64], habitat monitoring, perimeter monitoring [65]). Combining a filter bank with a template-based classifier allows the system to be used for any scenario where events can be distinguished from other events/noise based on the instantaneous frequency content. Further improvements on this acoustic processor could be gained by using more sophisticated classifiers (e.g. [82–84]) that have memory, and also by using signal features other than the spectrum, such as the cepstrum, which has been shown to provide better separation between acoustic classes for both speech and vehicle applications, and has been previously implemented in analog ICs [31]. If the application is changed to something that is not suitable for frequency analysis (e.g. imaging or chemical sensing), or if the domain is changed to something other than event detection (e.g. object localization/tracking), then a different set of operations will need to be implemented in the ASP.

Digital processing is likely to provide a simpler solution for systems in which a particular sensor node demands very high resolution processing, branching (e.g. state machines), long-term data storage of sensor information, or a high degree of flexibility for in-the-field reconfiguration. However, ASPs can still be used to complement the digital processors in such scenarios. For example, a system which requires high-resolution processing (e.g. SNR $>$ 16 bits) can use a lower-resolution ASP (where ASPs are always more efficient than digital circuitry [39]) to act as an energy-management tool to wake up a high-resolution digital system. Also, operations which require branching can be accomplished by incorporating state machines into the "Logic" portion of the ASP [see Fig. 3.2(b)]. Additionally, recent developments in programmable/reconfigurable analog systems enable analog integrated circuits to be general-purpose and easy to use, thereby providing significant flexibility and reducing the design time [35,85].

3.7 Conclusions

In this Chapter, we have described how ultra-low-power analog circuitry can be integrated with sensor nodes to reduce the node-level power consumption. We have shown the ability to interface these circuits with existing sensor platforms and have presented demonstrations to illustrate how analog hardware can reduce node resource usage and increase performance. We have implemented a vehicle classification system using our ASP-interfaced mote and have shown that it significantly improves the energy-efficiency over that of an all-digital implementation while retaining high classification accuracies.

We have utilized the strong points of both analog and digital such that each computational domain compensates for the limitations of the other. Specifically, by combining the ASP and the mote's microcontroller, we retain the flexibility of configuring system parameters at run-time and of implementing additional high-level decision making on the motes. At the same time, the use of the ASP enables ultra-low-power operation by reducing the amount of time that the mote is powered on and by reducing the required computational

resource implemented by the mote. By using both the analog and digital systems together, we have increased the lifetime of a wireless sensor network system from a few months to several years.

In the remainder of this Dissertation, we improve upon our "ASP-augmented mote" paradigm with the development of improved analog processing blocks (Chapters 4 and 5) and with the development of a low-overhead architecture for programming analog memory (Chapters 6 through 8). These developments are included in an improved version of the Hibernets processor (Chapter 9), as well as in a more capable and more flexible field-programmable analog array (FPAA) architecture (Chapter 10).

Chapter 4

A Low-Power and High-Precision Programmable Analog Filter Bank

Analog filter banks befit remote audio- and vibration-sensing applications, which require frequency analysis to be performed with low power consumption and with moderate to high precision. The precision of a filter bank depends on both the signal-path precision (i.e., dynamic range) and also the parameter precision (e.g., accuracy of the center frequencies). This work presents a new bandpass filter for audio-frequency filter banks and provides a procedure for designing this filter. The filter is used in a 16-channel filter bank which has been fabricated in a 0.35μm CMOS process. This filter bank has a dynamic range exceeding 62dB and consumes only 63.6μW when biased for speech frequencies. The filter bank's parameters are set via floating-gate current sources.[1] This work shows how to use these floating gates to obtain a versatile filter bank that can be precisely reprogrammed to arbitrary filter spacings and frequency weightings, with a parameter accuracy exceeding 99%.

The work in this Chapter was published in the IEEE Transactions on Circuits and Systems II [86]. Variations of the filter bank described in this Chapter have been a major component in all of our wireless-sensing-oriented analog processors, namely our Hibernets event detectors in Chapters 3&9 and our Netamorph field-programmable analog arrays in Chapter 10.

4.1 Analog Filter Banks

Due to the dynamic vibrational nature of many phenomena, sensor-processing applications often decompose signals into a time-frequency representation for analysis, manipulation, or recognition of the signal. Time-frequency representations characterize changes in the signal's spectrum over time and are typically implemented in two main ways: 1) a constant-bandwidth representation such as the short-time Fourier transform, which provides the most compact representation for signals with long-term periodicity and 2) a constant-relative-bandwidth or scale-space representation such as the wavelet transform, which provides the most compact representation for signals with localized time-frequency components [87].

[1] More details on floating gates are provided in Chapter 6.

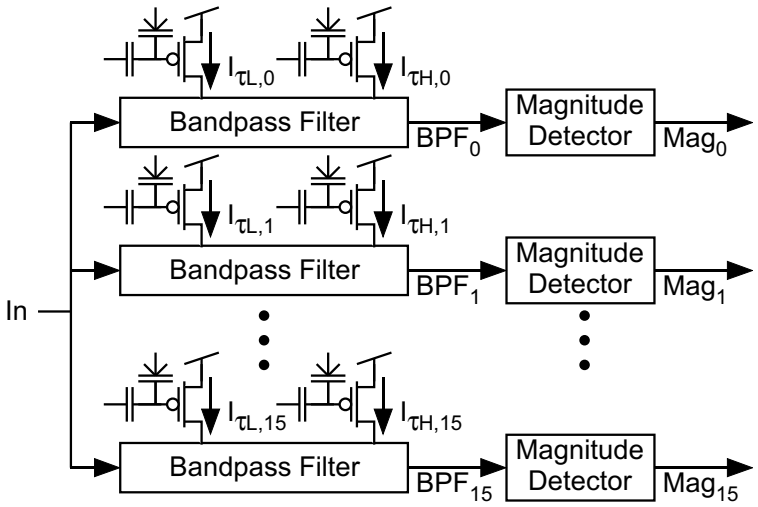

Figure 4.1: Block diagram of our analog filter bank chip, which combines a parallel array of bandpass filters with subband magnitude detectors. Each circuit parameter is controlled independently via floating-gate current sources, thus achieving precise control of the filter bank's settings.

Time-frequency analysis can be performed efficiently with analog filter banks [88], which naturally yield a scale-space representation [89]. Integrated analog filter banks were incubated in the research of silicon cochleae [90–94], and their low-power operation makes them well suited to battery-powered applications such as cochlear implants [95, 96] and wireless sensor networks [41].

In order for analog filter banks to be an option for incorporation into mature systems, they should operate with high precision, and their filter parameters should be reprogrammable. The precision of a filter bank is affected by both the signal-path precision (e.g., dynamic range) and the parameter precision (e.g., center frequencies and bandwidths). A large dynamic range is difficult to achieve in low-power circuits due to the relatively high noise levels and reduced voltage headroom [93]. Precise tuning of filter parameters is difficult to achieve in compact and low-power circuits due to process variations, which can be partially overcome by using large device sizes and additional circuitry. Nevertheless, state-of-the-art filter banks have center-frequency errors >10% [91]. Previous filter banks have been demonstrated to achieve either a large dynamic range [96] or precise and reprogrammable parameters [97], but in this work, we present a filter bank which achieves both a large dynamic range and is also precise and programmable, while maintaining low power consumption.

Figure 4.1 shows the block diagram of our 16-channel filter bank chip, which was fabricated in a 0.35μm standard CMOS process. The filter bank performs time-frequency analysis with a parallel array of bandpass filters. For the bandpass filter, we use our new OTA-based (operational transconductance amplifier) capacitively-coupled current conveyor (C^4), which significantly improves upon the shortcomings of the transistor-based version [79]. In Section 4.2, we present this bandpass filter, demonstrate its large dynamic range, and provide a design procedure. To achieve precise programmability at low-power operation, we set the filter

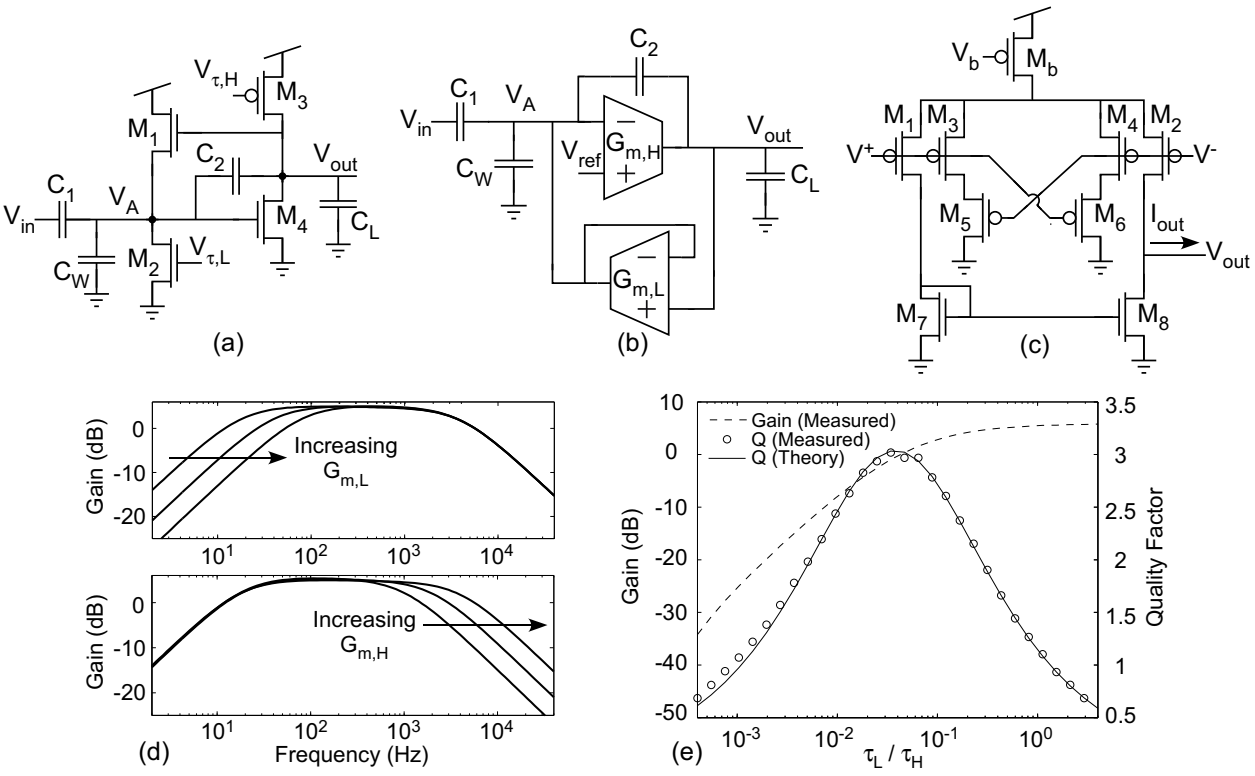

Figure 4.2: (a) The original transistor-based C^4. (b) The new OTA-based C^4. (c) The symmetric "bump"-linearized transconductor that is used in the OTA-C^4. (d) Measured frequency responses of the OTA-C^4. The independence of the corner frequencies is demonstrated by stepping one bias at a time. (e) The measured gain and Q of the OTA-C^4, both plotted as a function of the time constant ratio τ_l/τ_h, which is proportional to $G_{m,H}/G_{m,L}$.

bank parameters with floating-gate-based current sources (i.e., non-volatile analog memory), which have been shown to be a good option for achieving precise and programmable biasing in CMOS circuits [97]. Through the use of floating-gate transistors, we have consistently achieved a percent error below 1% when programming the center frequencies, gains, and bandwidths of the filters. In Section 4.3, we demonstrate the precise programmability of the filter bank by programming it to different filter spacings and frequency weightings. All plots in this Chapter are measurements from the fabricated filter bank.

4.2 Bandpass Filter

Transconductance-capacitance (G_m–C) topologies are a common choice for low-power integrated filters. When low-power operation is required, G_m–C filters can be operated in the subthreshold domain. Since OTAs have low transconductance in the subthreshold region, these filters are able to achieve the long time constants that are needed for audio-frequency operation while using small integrated capacitors, thus enabling compact designs. G_m–C

filters are also good for applications requiring programmability since the transconductance values, which are easily controlled by adjusting OTA bias currents, appear in the transfer function as parameters of the filter (thus the filter parameters are easily modified). Furthermore, since these filter parameters typically scale with center frequency, filter bank spacing is easily achieved with current ratioing or resistive dividers. For these reasons, most low-power filter banks have been based on G_m–C filters.

Unfortunately, the dynamic range of G_m–C filters is limited by the small linear range of the subthreshold differential pair. This problem is often mitigated through transconductor linearization [98, 99] or through capacitive division to keep the signal within the linear range of the transconductors [100]; we use both of these techniques in this work. Another limitation to the dynamic range of low-power G_m–C filters is the small subthreshold current, which results in relatively high noise levels for the bandwidth, and can be alleviated by increasing both the capacitor sizes and the power level.

4.2.1 OTA-C^4 Bandpass Filter[2]

The bandpass filter that we created for our filter bank is our new OTA-based capacitively-coupled current conveyor (C^4) shown in Fig. 4.2(b). This filter is based on the previously reported transistor-based C^4 [79], which is shown in Fig. 4.2(a). The C^4 filter topology offers flexible tuning, with run-time adjustable center frequency, gain, and quality factor (Q). In the OTA-based version of the C^4, the high-gain inverting transconductor $G_{m,H}$ replaces the common-source amplifier M_3–M_4, and the follower-configured transconductor $G_{m,L}$ replaces the source follower M_1–M_2. We have developed the OTA-C^4 to more easily increase the filter's linear range and also to obtain control over the filter's DC operating point.

In the OTA-C^4, capacitors C_1 and C_W form a capacitive divider that attenuates the AC input onto the central node V_A and makes the filter's response independent of V_{in}'s DC level. The negative feedback around $G_{m,H}$ holds V_A at a DC level of V_{ref}. In the transistor-based C^4, the source follower causes an offset in the feedback path which can shift the equilibrium point away from the center of the inverting amplifier's linear range, reducing the filter's linear range below the linear range of its transconductance elements. The OTA-C^4 fixes this by feeding back through a low-offset follower OTA ($G_{m,L}$).

Another problem with the transistor-based C^4 is that the common-source amplifier's quiescent point is bias-dependent; this means that the output DC level of each filter in the array will be different, thereby requiring some way of correcting for these differences in order to compare/combine the outputs of different channels. The OTA-C^4 fixes this problem by using the non-inverting terminal of $G_{m,H}$ to globally set the DC level of the filter bank.

Additionally, the use of OTAs makes the filter modular, offering the designer the flexibility to optimize the filter for their application, e.g., further extending the linear range or reducing the transconductance for ultra-low-frequency applications. We have extended the linear range of the OTA-C^4 by using the symmetric "bump" OTA shown in Fig. 4.2(c) [99], which has four times the linear range of the standard differential pair.

[2] A detailed analysis of the OTA-C^4 is provided in Appendix C.

The transfer function for the OTA-C^4 is

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{s\tau_l\left(1-s\tau_f\right)}{1+s\left(\tau_l+\tau_f\left(\frac{C_O}{C_2}-1\right)\right)+s^2\tau_h\tau_l} \tag{4.1}$$

where $C_T = C_1 + C_2 + C_W$ and $C_O = C_2 + C_L$ and

$$\tau_l = \frac{C_2}{G_{m,L}}; \quad \tau_h = \frac{C_O C_T - {C_2}^2}{C_2 G_{m,H}}; \quad \tau_f = \frac{C_2}{G_{m,H}} \tag{4.2}$$

τ_l is the time constant of the low corner frequency and τ_h is the time constant of the high corner frequency. These time constants are controlled independently by the transconductances $G_{m,L}$ and $G_{m,H}$, respectively, as shown in Fig. 4.2(d). Proper capacitor sizing ensures that the feed-through time constant τ_f is at a sufficiently high frequency such that its effect on the numerator of the transfer function can be ignored, so that the transfer function takes the familiar form of a bandpass filter.

The OTA-C^4 is a flexible bandpass filter, with run-time tunable center frequency, gain, and Q, which are all established via the transconductances. The gain and Q both depend on the ratio $G_{m,H}/G_{m,L}$, and are specified as

$$|A_v| = \frac{C_1}{C_2}\frac{1}{1+\frac{C_L}{C_2}\frac{G_{m,L}}{G_{m,H}}}; \quad Q = \frac{\sqrt{C_T C_O - C_2^2}}{C_L\sqrt{\frac{G_{m,L}}{G_{m,H}}}+C_2\sqrt{\frac{G_{m,H}}{G_{m,L}}}} \tag{4.3}$$

Figure 4.2(e) shows the measured gain and Q of the OTA-C^4 for different transconductance ratios as a function of τ_l/τ_h, which is proportional to $G_{m,H}/G_{m,L}$. Equation (4.3) shows that the gain is highest when $G_{m,H} \gg G_{m,L}$ where it approaches a value of C_1/C_2, which is 2 (6dB) for this particular implementation. As $G_{m,H}/G_{m,L}$ decreases, the corner frequencies cross and cause the gain to decrease. The Q has a maximum value when $G_{m,H}/G_{m,L} = C_L/C_2$ and decreases symmetrically as the transconductance ratio changes. When biased for maximum Q, the gain is $|A_v| = C_1/(2C_2)$.

4.2.2 Design

We have developed an algorithmic design procedure for the OTA-C^4 that is similar to the procedure for the transistor-C^4 in previous work [79] but that has been modified for this new circuit. This design procedure helps the designer to choose the device sizes that are needed to achieve the desired dynamic range (DR) in decibels, maximum gain ($A_{v,max}$), and maximum Q (Q_{max}), as well as to choose the currents that are needed to achieve the desired filter characteristics. We make the following assumptions for this procedure: 1) the dynamic range requirement is met at the Q_{max} condition, 2) the maximum output amplitude ($V_{out,max}$) is equal to the linear range of the transconductors (V_L), and 3) the transconductor noise is mostly white, as is typical for subthreshold OTAs [98].

1. Choose C_2 to meet the DR specification.

$$C_2 = \frac{Nq}{4V_L}10^{\frac{DR}{10}} \tag{4.4}$$

where N is the number of noise sources in the transconductors [98] and q is the charge of an electron.

2. Choose C_1 either for (a) the desired maximum gain or (b) the desired gain at the Q_{max} condition.
$$C_1 = \begin{cases} C_2 A_{v,max} & (a) \\ 2C_2 A_{v,Q} & (b) \end{cases} \tag{4.5}$$

3. Choose C_T for linearity: $C_T = 4C_2 Q_{max}^2$

4. C_L should be on the same order as C_T for step 1 to be valid but can otherwise be used to position the feed-through time constant (τ_f) or to optimize for power.
$$P = \frac{C_L}{V_L} V_{dd} Q_{max} f_c \tag{4.6}$$

5. Use Equation (4.3) to choose the transconductance ratio ($R = G_{m,H}/G_{m,L}$) for either the desired gain or Q.

6. Choose the transconductance values for the desired center frequency (f_c) using
$$G_{m,L} = \sqrt{\frac{C_O C_T - C_2^2}{R}} 2\pi f_c; \quad G_{m,H} = R G_{m,L} \tag{4.7}$$

4.2.3 Performance

Using the above design procedure, the filters for our filter bank chip were designed for a dynamic range of 60dB, a maximum gain of 6dB, and a maximum Q of 3. The frequency response measurements in Fig. 4.2(d)–(e) verify that the filter meets the maximum gain and maximum Q specifications.

To verify that the fabricated filter meets the dynamic range specifications, we determined the maximum and minimum signal levels while the filter was simultaneously biased for unity gain and maximum Q. The experiments were performed at four center frequencies across the audio frequency range (20Hz, 200Hz, 2kHz, and 20kHz). The minimum signal level is defined as the integrated output noise and the maximum signal level is defined as the output-referred 1dB compression point. The compression point is an appropriate criterion for spectral analysis applications, for which only the magnitude of the filter output is required.

The circuit was designed for a dynamic range that extends from $70.7\mu V_{rms}$ to $70.7mV_{rms}$. The measured output-referred noise is shown in Fig. 4.3(a). The integrated output noise values are all well below the designed noise floor and are listed in Table 4.1 along with all other measured performance results. Figure 4.3(b) shows the compression point measurements, which exceed $70.7mV_{rms}$ for all cases (a 2.5-times improvement over the original C^4 [79]), yielding a dynamic range of over 62dB for all four frequencies. To further characterize the linearity, we also measured the total harmonic distortion (THD) and the intermodulation distortion (IMD). The THD was measured to be less than 1% for $70.7mV_{rms}$ inputs and the OIP3 point was measured to be $141.4mV_{rms}$, or -4dBm (50Ω reference).

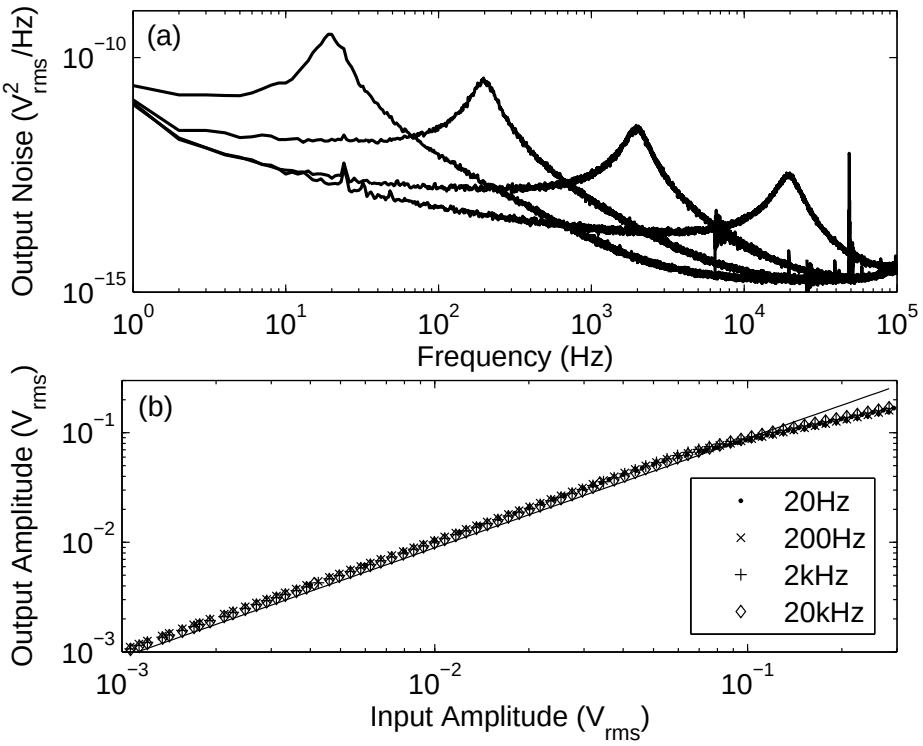

Figure 4.3: (a) Measured output-referred noise of the OTA-C^4. (b) Compression point measurement for the OTA-C^4. The line shows the 1dB deviation.

Table 4.1: Performance Results

Metric	20Hz	200Hz	2kHz	20kHz
Noise (μV_{rms})	59.8	59.7	58.2	58.7
1dB Comp. Point (mV_{rms})	76.5	80.3	87.4	86.8
1dB Comp. Point (dBm)	-9.32	-9.00	-8.16	-8.22
1dB DR (dB)	62.1	62.6	63.5	63.4
THD at 70.7mV_{rms}	0.89%	0.89%	0.84%	0.61%
OIP3	141.4mV_{rms}; -4.00dBm			
Power	19.0nW	198nW	2.85μW	75.4μW

In Table 4.2, we compare the OTA-C^4 with other recently reported, low-power, second-order, voltage-mode, audio-range, bandpass filters. We note that three of the filters are fully differential [102–104] and all but one of the topologies [100] in the Table are capable of complex poles (i.e. Q >0.5). In comparison to the other filters, the OTA-C^4 achieves comparable performance and is compact with the fewest transconductors.

Table 4.2: Comparison Amongst Low-Power, Second-Order, Audio-Range, Bandpass Filters

	Technology	Freq. Range	DR	Power
Proposed	0.35μm	20Hz–20kHz	62.1–63.5dB	19nW–75.4μW
[100]	1.5μm BiCMOS	100Hz–10kHz	62dB	2nW–2μW
[101]	0.5μm	700Hz–4kHz	55dB	41.12μW (1kHz)
[102]	0.35μm	30Hz–30kHz	51dB	290nW (660Hz)
[103]	0.35μm	100Hz–30kHz	62.3dB	<16μW
[104]	0.8μm	100Hz–2kHz	62–78dB	2.5μW (2kHz)

4.3 Filter Bank

Our filter bank chip has been fabricated in a 0.35μm CMOS process and is shown in Fig. 4.4(f). The filter bank has 16 parallel channels, with each channel consisting of an OTA-C[4] and a magnitude detector, as shown in Fig. 4.1. Details of the magnitude circuit are provided in Chapter 5. In order to achieve precise programmability, we use floating-gate transistors to bias the circuits [97]; this configuration is depicted in Fig. 4.1. Floating-gate transistors are MOSFETs that have only capacitive inputs to their gates, a structure which can be formed in standard CMOS processes. The amount of charge on the floating gate can be programmed via Fowler-Nordheim tunneling and hot-electron injection. This programmed charge is non-volatile and provides fine control over the drain current. In order to control $G_{m,H}$ and $G_{m,L}$ using the floating-gate transistors, the floating-gate currents are copied into transistor M_b of the OTA-C[4] transconductors using both an nFET and a pFET current mirror.

4.3.1 Filter Characterization and Programming

Through the use of programmable biasing, we are able to correct for fabrication mismatch and process variations, enabling us to set the circuit parameters with very high accuracy. To correct for these variations after fabrication, we performed the following characterization routine, which has three steps. First, I_H ($G_{m,H}$'s bias current) is stepped, and its associated time constant (τ_h) is measured at each step to determine the mapping between I_H and τ_h independent of component tolerances. During this step, I_L ($G_{m,L}$'s bias current) is held at a low value so that the low corner frequency does not interfere with measuring the high corner frequency (see Fig. 4.2(d)). Second, I_H is held at a high value (so that the high corner frequency has no effect on measurements) and I_L is stepped to determine its mapping to τ_l. Third, the ratio of τ_l/τ_h is swept, this time measuring the Q and gain, which are fit to the expressions in (4.3). This routine can be optimized for speed by measuring a small number of well-chosen data points. Characterization only needs to be performed once for each filter to successfully cancel out fabrication mismatch.

Once a filter has been characterized, the last two steps of the design procedure are used to determine the currents required for the desired combination of f_c and A_v or f_c and Q. The floating gates are then programmed to these currents [105] to achieve the desired filter characteristics. Our programming algorithm, which has not been optimized for speed, takes

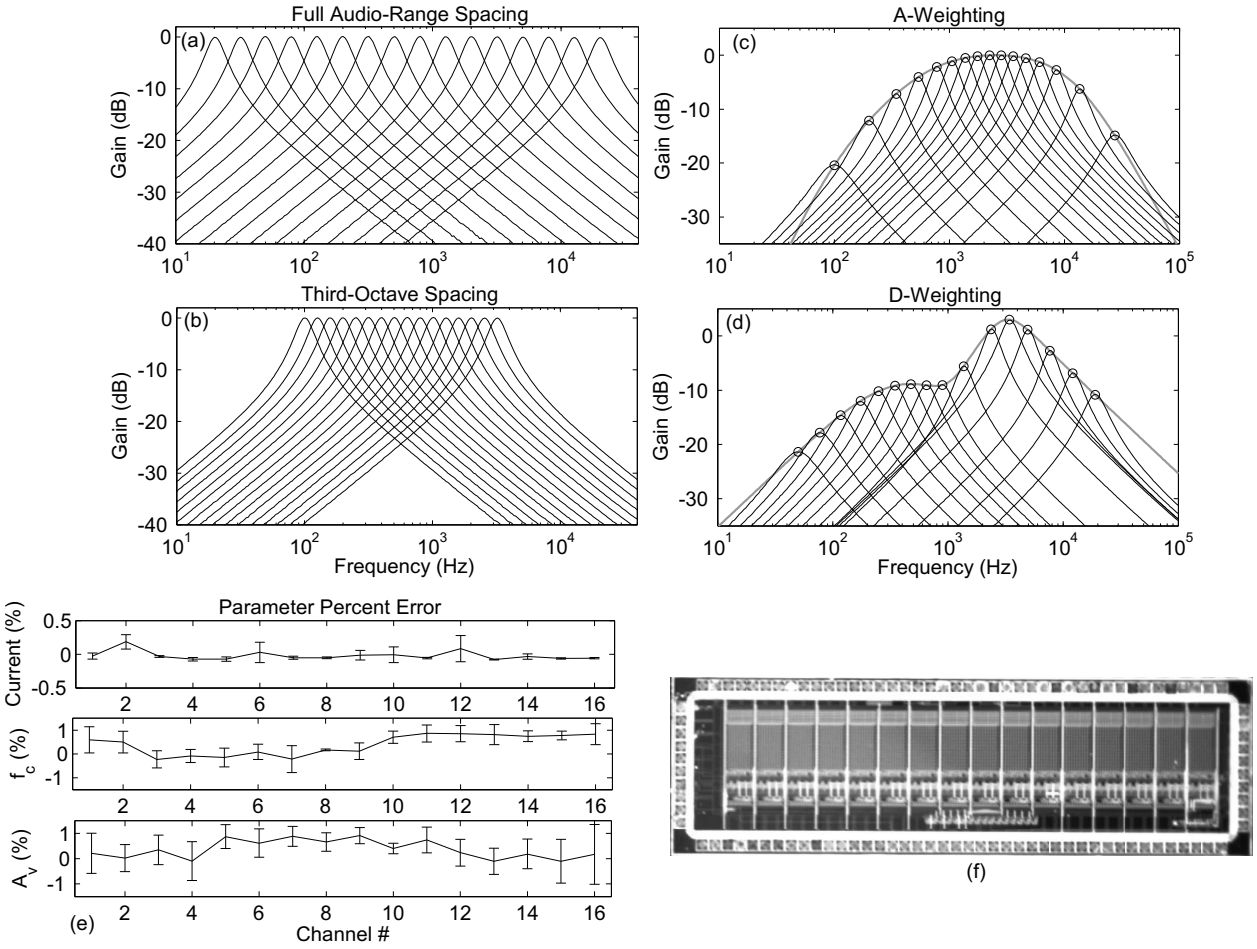

Figure 4.4: Measured filter bank AC responses. (a) Biased for 20Hz–20kHz with unity gain. (b) Biased for unity gain and third-octave spacing starting at 100Hz. (c) A-weighting. (d) D-weighting. The circles show the targeted gains and center frequencies. (e) Percentage error results of the four filter bank biasings in parts (a)–(d). The mean and standard deviation of the percentage error are shown for the currents, center frequency, and gain of each channel. (f) Die photograph of our filter bank chip. The chip is 4.8mm x 1.4mm. The dimension of a single OTA-C[4] is 560μm x 237μm.

approximately 30 seconds per gate, and program times on the order of milliseconds have previously been reported with similar algorithms [105]. Because of the independent control of these filter parameters, the filter bank is very versatile. Independent control of each filter's center frequency and Q enables the user to program the filter bank to cover the frequency range of interest or even to focus on specific frequencies if signal characteristics are known *a priori*. Independent control of the gain of each channel allows the user to selectively emphasize bands or to adopt common perceptually-derived frequency weightings, such as A- or D-weighting, without requiring a multiplier in each band.

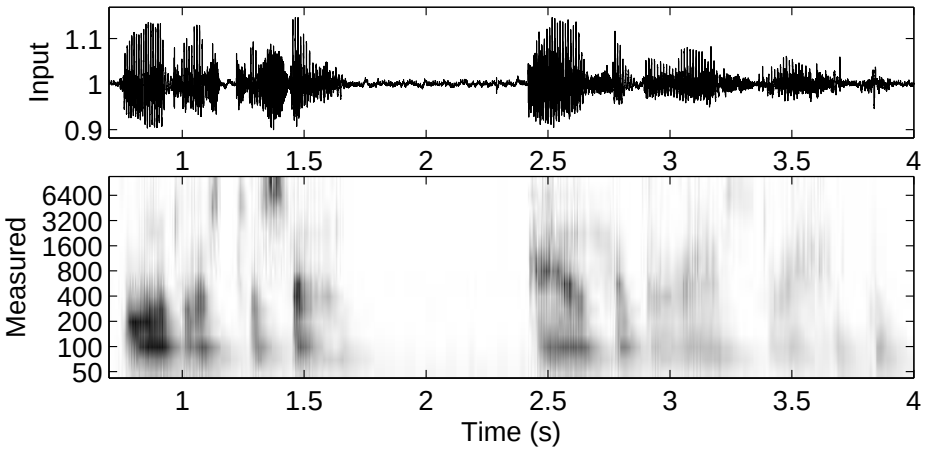

Figure 4.5: Time-frequency analysis performed with the filter bank. The top subplot is the speech waveform which was streamed into the filter bank. The bottom subplot is the resulting spectral decomposition.

4.3.2 Demonstrations

To verify the accuracy of the parameter programming and also to demonstrate the versatility of the filter bank, we programmed the filter bank to the following filter spacings and frequency weightings: 20Hz–20kHz with unity gain, third-octave spacing with unity gain, A-weighting, and D-weighting. The resulting measured AC responses are shown in Fig. 4.4(a)–(d). The measured accuracies of the programmed currents and parameters in Fig. 4.4(a)-(d) are shown in Fig. 4.4(e). The mean absolute percentage error across all channels is 0.087% for the currents, 0.536% for center frequency, and 0.634% for gain.

To demonstrate the filter bank performing time-frequency decomposition, we streamed a speech waveform through the filter bank, which was biased for half-octave spacing. A simulated version of the magnitude circuit was used to extract the magnitude of the measured output waveforms of the filter bank. The resulting spectral magnitude is shown in Fig. 4.5.

4.4 Conclusion

We have presented a low-power and programmable analog filter bank that achieves both high signal-path precision (>62dB) and high parameter accuracy (>99%). Thus, this filter bank meets the requirements for inclusion in today's demanding battery-powered audio-processing systems. Furthermore, we have illustrated the utility of using floating-gate transistors for precise, programmable, and low-power systems by demonstrating the high accuracy which can be achieved when programming arbitrary array settings.

Chapter 5

A Low-Power Magnitude Detector for Analysis of Transient-Rich Signals

Magnitude detection, such as envelope detection or RMS estimation, is needed for many low-power signal-analysis applications. In such applications, the temporal accuracy of the magnitude detector is as important as its amplitude accuracy. We present a low-power audio-frequency magnitude detector that simultaneously achieves both high temporal accuracy and high amplitude accuracy. This performance is achieved by rectifying the signal with a high-ripple peak detector and then averaging this rectified signal with an adaptive-time-constant filter. The time constant of this filter decreases with increasing amplitude, enabling the filter to quickly respond on a short time scale to transients, while steady-state ripple is averaged on a longer time scale. The circuit has been fabricated in a 0.18μm CMOS process and consumes only 1.1nW–1.08μW when tuned for operation from 20Hz–20kHz. It exhibits a dynamic range of 70dB across typical speech frequencies.

The work in this Chapter was published in the IEEE Journal of Solid-State Circuits [106]. Variations of the magnitude detector described in this Chapter have been a major component in all of our wireless-sensing-oriented analog processors, namely our Hibernets event detectors in Chapters 3&9 and our Netamorph field-programmable analog arrays in Chapter 10.

5.1 Magnitude Detector Circuits

Magnitude-detection circuits—such as envelope detectors, peak detectors, and RMS-to-DC converters—produce an estimate of a signal's magnitude and are thus important elements in communications transceivers [107], automatic gain control systems [108], and analog spectral analyzers [27, 90–92, 97]. Since the magnitude is a time-varying quantity, the accuracy of a magnitude detector has two components: amplitude accuracy and temporal accuracy. Traditionally, the design emphasis of magnitude circuits has been on amplitude accuracy; however, temporal accuracy is crucial when the magnitude changes quickly relative to the frequencies of the underlying carrier signal, such as in speech signals [109]. Thus, temporal accuracy is important in audio-processing systems, including ultra-low-power (ULP) applications such as bionic ears [27, 95] and event detectors for wakeup applications [41]. Existing digital signal processing techniques for temporal accuracy use non-physical, non-

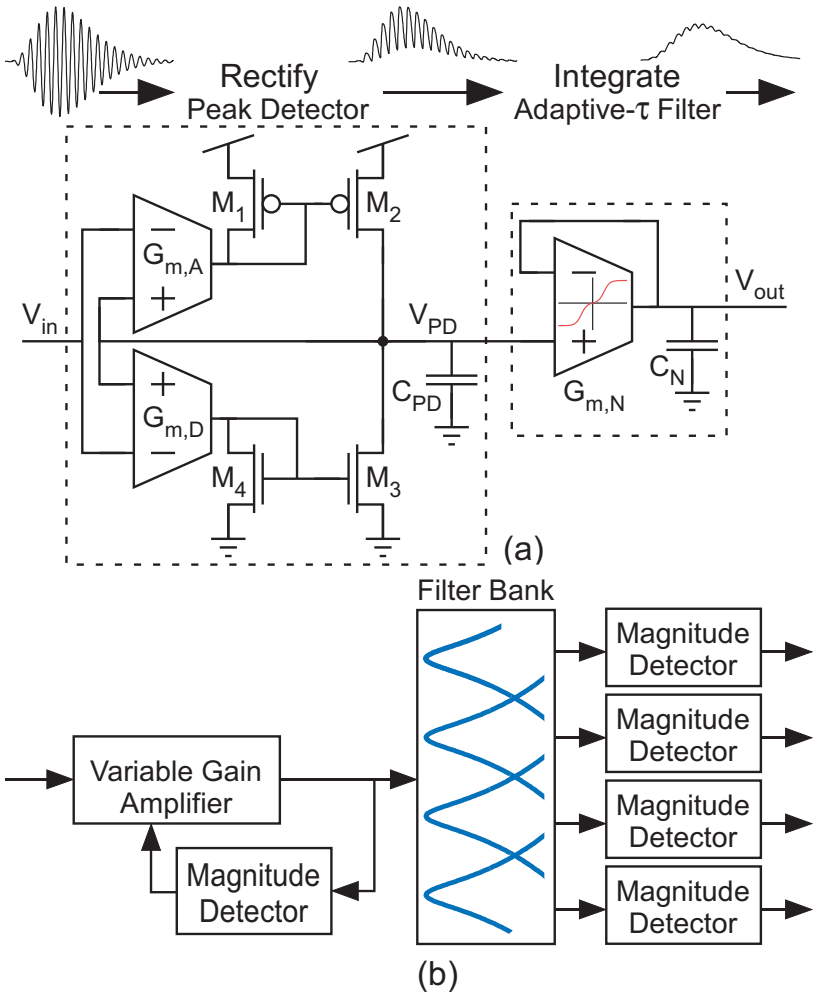

Figure 5.1: (a) Schematic of our magnitude detector. (b) Block diagram of an analog spectral-analysis system.

causal filters that require powerful processing and non-stop data conversion, thereby limiting their use in ULP systems. Discrete-time analog circuits have also been explored to achieve good temporal accuracy [110]; however, the power level and the sampled-data representation are inappropriate for many ULP signal-analysis systems. In this work, we present an ULP continuous-time magnitude detector that has been designed with an emphasis on temporal accuracy, while still achieving high amplitude accuracy.

Figure 5.1(a) shows our magnitude detector, wherein a rectifying nonlinearity provides an initial estimate of the signal's magnitude and then a lowpass filter averages this estimate to obtain the final smooth magnitude estimate. For the rectifying nonlinearity, we have developed a voltage-mode asymmetric integrator. This circuit's asymmetry causes the average level of its output to shift in proportion to the input magnitude, thereby providing a magnitude estimate that is superimposed with a ripple. This ripple is then smoothed by a nonlinear lowpass filter with an adaptive time constant. We have designed this filter's nonlinearity such that its time constant shrinks in response to large input-output differ-

ential signals, thereby reducing the integration window in order to follow transients more closely, while maintaining a long time constant for small signals in order to retain good ripple suppression.

The remainder of this Chapter is organized as follows. In Section 5.2, we discuss the application-space of our circuit and describe the high-level design approach. In Sections 5.3 and 5.4, we present the asymmetric integrator and the adaptive-time-constant filter, respectively. Then, in Section 5.5, we combine the subcircuits into the complete magnitude detector and present our experimental results. We have fabricated this circuit in standard 0.18μm, 0.35μm, and 0.5μm CMOS processes. Unless otherwise noted, all plots are measurements from the 0.18μm circuit.

5.2 Magnitude Detector Architecture

A common application environment for magnitude detectors is within audio- and vibration-processing systems. For example, a standard first step in such systems is spectral analysis, which can be implemented in low-power analog circuits to make an efficient real-time sensor-processing front-end [29, 59, 111]. Such a spectral analysis front-end can be combined with other analog processing circuits to create an entire ULP system (such as for implantable electronics), or the front-end can be used as an event detector to wake up a higher-power back-end, thus reducing system-level power consumption [41]. Analog spectral-analysis systems typically consist of a bank of filters that decompose the signal into frequency components [27, 90–92] followed by subband processing blocks, such as magnitude detectors for extracting the magnitude of the spectrum [112–114], as illustrated in Fig. 5.1(b). Adaptation is also often used to increase the dynamic range of the system, either by using automatic gain control on the pre-filtered signal [27] or by adapting the gain/Q in individual subbands. Such adaptation is typically based on the signal's magnitude, and therefore requires a magnitude detector. In addition to needing to respond quickly to changes in the signal, magnitude detectors for these applications require a smooth/low-ripple magnitude estimate, since ripple adds uncertainty to the estimate.

Magnitude detectors for these types of applications are typically peak/envelope detectors which extract the envelope of the waveform by finding local maxima and then providing a slow decay between individual peaks. This operation is illustrated in Fig. 5.2(a), which shows numerical simulations of a peak detector responding to a speech input. As can be seen in Trace (i), a peak detector with a slow decay provides a smooth envelope of the waveform. However, the slow decay rate causes the peak detector to respond too slowly to decreases in the input's magnitude, thereby masking low-amplitude content that follows high-amplitude content. This temporal masking loses information about the signal, which is unacceptable for analysis applications. Setting a faster decay rate, as shown in Trace (ii), enables the peak detector to track the decreases in amplitude, but the output has too much ripple to be useful as a magnitude estimate. Consequently, a tradeoff exists between a smooth/low-ripple response and being able to quickly track signals so that information is not lost. To address this tradeoff, we have developed an adaptive-time-constant lowpass filter that is able to operate on a high-ripple output of a peak detector (similar to Trace (ii)) to provide a low-ripple output with good temporal accuracy, as shown in Trace (iii). This

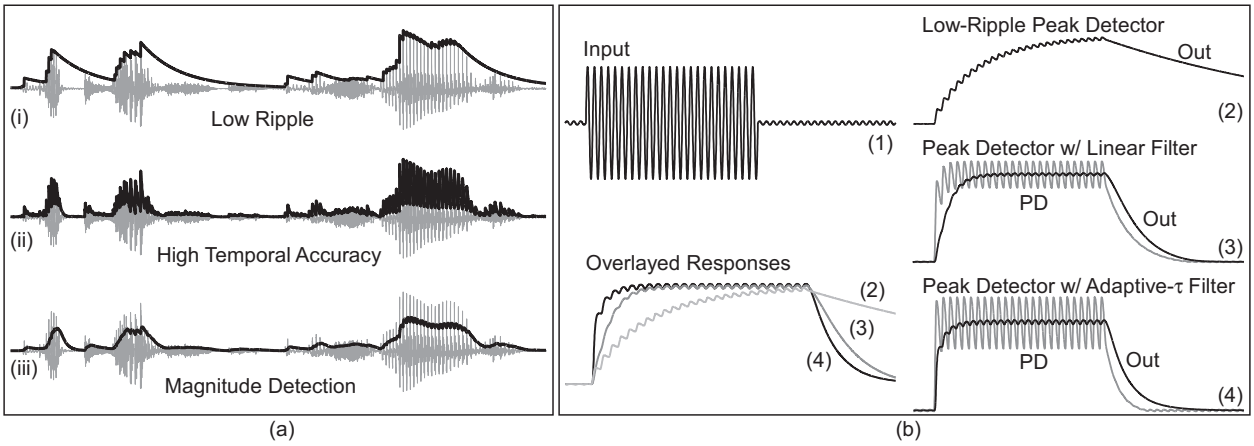

Figure 5.2: (a) Tradeoff between a response with low ripple and a response with high temporal accuracy. (b) Comparing the temporal accuracy of three magnitude-detection architectures, each with 1% ripple. "Overlayed Responses" shows the improved temporal response achieved by our adaptive-time-constant filter. All plots in this Figure are from numerical simulations of the equations discussed in Sections 5.3 and 5.4.

filtered output is a scaled version of the envelope, prompting the term magnitude detector instead of peak/envelope detector.

To illustrate that an adaptive-time-constant filter is needed to simultaneously achieve a low-ripple output and a good temporal response, we provide the numerical simulations of Fig. 5.2(b). This Figure demonstrates the "acquire times" for different magnitude architectures, each of which provide 1% ripple. Low-ripple operation can be achieved with a standalone peak detector by slowing down its operation and using its inherent integration; however, the resulting response has a long acquire time and responds slowly to changes in the envelope of the input, as shown in Trace (2). The acquire time can be improved by cascading a lowpass filter with a peak detector, where the peak detector is biased to respond quickly to rising/falling signals and the lowpass filter is used to smooth the ripple; this response is shown in Trace (3). While the acquire time is improved for the downward step, the upward step response is limited by the filter's time constant, which is longer than the peak detector's attack time constant in order to achieve low ripple. To improve the acquisition time while still achieving low ripple, we need a filter that adjusts its time constant based on the amplitude of the signal. To accomplish this, we have developed a nonlinear filter with an adaptive time constant. When the amplitude changes, the integration window shrinks to track more quickly; when in the steady state, the time constant returns to a larger value to suppress the ripple. This operation is demonstrated in Trace (4) where the increasing amplitude is followed more quickly than for the linear filter in Trace (3). This nonlinearity in the filter helps the magnitude circuit to achieve better temporal responsiveness and still achieve low ripple in the steady state.

In summary, our magnitude detector (Fig. 5.1(a)) consists of an asymmetric integrator followed by an adaptive-time-constant filter. In the remainder of this Chapter, we describe these two subcircuits in detail and show the results of the complete magnitude detector

circuit.

5.3 Peak Detector

In this Section, we present a voltage-mode peak detector that provides an initial estimate of a signal's magnitude. This circuit has tunable attack and decay integration rates, allowing it to be used as an asymmetric integrator. By setting the attack rate faster than the decay rate, the average level of the output shifts in proportion to the amplitude of the input signal, thereby providing a measure of the signal's magnitude. Due to its tunability, this circuit can be biased to extract any fraction of the input envelope (such as the full envelope or $1/\sqrt{2}$ for RMS detection) and with any amount of ripple. In this Section, we describe the development of this peak detector circuit, analyze its operation, and provide a design procedure to allow it to be used in the larger magnitude-detector circuit.

5.3.1 Overview of the Peak Detector Circuit

Figure 5.3(a) shows a common voltage-mode CMOS peak detector topology [115–117] that is based on a peak-detect-and-hold circuit [118] but with the reset transistor replaced by the constant-current sink, M_3. In this circuit, the current mirror half-wave rectifies the output current of the operational transconductance amplifier (OTA) onto the capacitor, causing the circuit to act as a follower when $V_{in} > V_{out}$ (i.e. the attack phase), assuming that the transconductance $G_{m,A}$ is large; reducing $G_{m,A}$ causes the circuit to act as a follower-integrator during the attack phase with an attack time constant $\tau_A = C_{PD}/G_{m,A}$. When $V_{in} < V_{out}$ (i.e. the decay phase), no current flows through M_2, and the capacitor discharges through M_3, causing the output voltage to decrease at a constant rate. The response of the circuit to downward steps of varying sizes is shown in Fig. 5.3(c), which illustrates that the circuit has a constant decay-rate regardless of the step size. Unfortunately, this circuit cannot be biased for both good dynamic range and good temporal performance. For example, a very slow decay rate must be used in order to detect small signals and provide some amount of "holding" instead of simply following the small downward steps (see the first downward step in Fig. 5.3(c)). However, using this same slow rate for large signals results in very long acquire times, thereby limiting the circuit's temporal response.

A precision magnitude detector, however, should produce a magnitude estimate that is amplitude invariant and has low temporal error. To accomplish these requirements, we have altered the peak detector of Fig. 5.3(a) to decay with a time constant, resulting in the circuit of Fig. 5.3(b). The operation of the lower half of the circuit mirrors that of the top half, causing the circuit to follow downward-going signals with time constant $\tau_D = C_{PD}/G_{m,D}$. The amplitude-dependent decay is demonstrated in Fig. 5.3(c), showing that it provides an appropriate decay over a much larger range of steps than the constant decay-rate version. The circuit is typically biased with $G_{m,A} > G_{m,D}$ to extract the positive magnitude, and it can be thought of as an asymmetric integrator due to its two different time constants, as described by the piecewise differential equation

$$C_{PD}\frac{dV_{out}}{dt} = \begin{cases} G_{m,A}(V_{in} - V_{out}), & V_{in} > V_{out} \quad (attack) \\ G_{m,D}(V_{in} - V_{out}), & else \qquad\quad (decay) \end{cases} \tag{5.1}$$

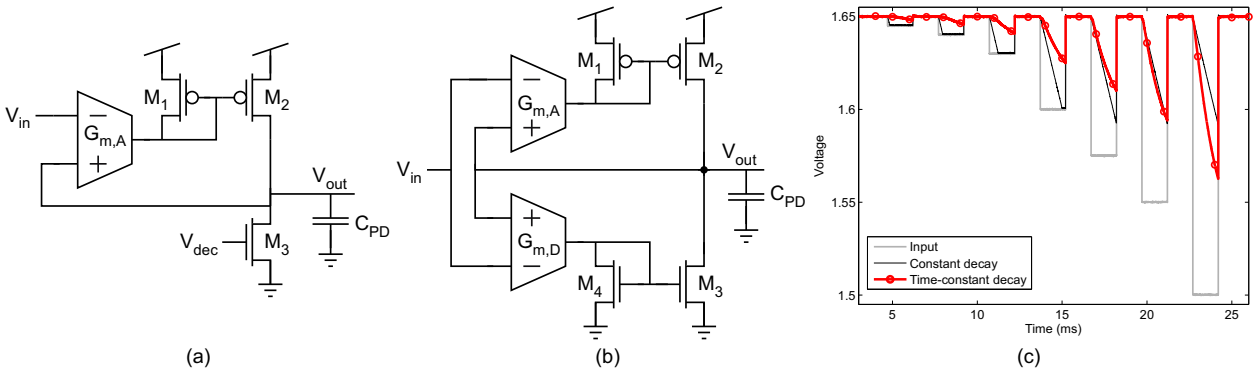

Figure 5.3: (a) Peak detector with a constant decay-rate. (b) Our peak detector with a time-constant decay. (c) Comparing the constant decay-rate and time-constant decay peak detectors. For a fair comparison, the measurements were taken from circuits fabricated on a 0.5μm CMOS process, since that is the only process in which we have fabricated the constant decay-rate peak detector.

5.3.2 Peak Detector Analysis[1]

This peak detector can be biased for different tracking levels and ripple levels through the choice of the transconductance values $G_{m,A}$ and $G_{m,D}$. We define the tracking level, $A_t = V_{out,DC}/V_{in,pk}$, as the DC output level normalized by the input amplitude. The tracking level is the magnitude metric; for example, $A_t = 1/\sqrt{2}$ is used for RMS tracking. We define the ripple ratio, $R_O = V_{out,pk}/V_{in,pk}$, as the amplitude of the output ripple normalized by the amplitude of the input. Typically, the peak detector is tuned for 10%–30% ripple to achieve 1% ripple from the complete magnitude circuit, as discussed further in Section 5.5. To design and bias this peak detector for use in the complete magnitude detector, we need to know how to choose the biases to achieve the specified tracking levels and ripple amounts.

To derive the dependence of both the tracking level and the ripple ratio upon the peak detector's biases, we have used the harmonic balance method [119, 120]. This procedure uses (5.1) modeled in the form of Fig. 5.4(a), and it assumes a sinusoidal input. First, the output equation is written in terms of both the tracking level and the output ripple, yielding $V_{out} = V_{in,pk}(R_O \sin(\omega t + \phi) + A_t)$, where $V_{in,pk}$ is the input amplitude and ϕ is the phase shift from the input to the output. Next, this equation is applied to (5.1), and the terms are balanced both at 0Hz and at the fundamental frequency to obtain equations for the tracking level and the ripple ratio, respectively. Since the peak detector has a lowpass form, we have neglected the harmonics of the fundamental to derive an approximation that is sufficient for choosing biases and predicting the operation of the circuit.

By solving the loop at 0Hz, the following equations were obtained for the tracking level, A_t

$$\frac{1}{2}\log\left(\frac{G_{m,A}}{G_{m,D}}\right) = \coth^{-1}\left(\frac{2}{\pi}\left(\frac{R_e}{A_t}\sqrt{1-\frac{A_t^2}{R_e^2}} + \sin^{-1}\left(\frac{A_t}{R_e}\right)\right)\right) \tag{5.2}$$

[1]More details on the analysis of the peak detector are provided in Appendix D.

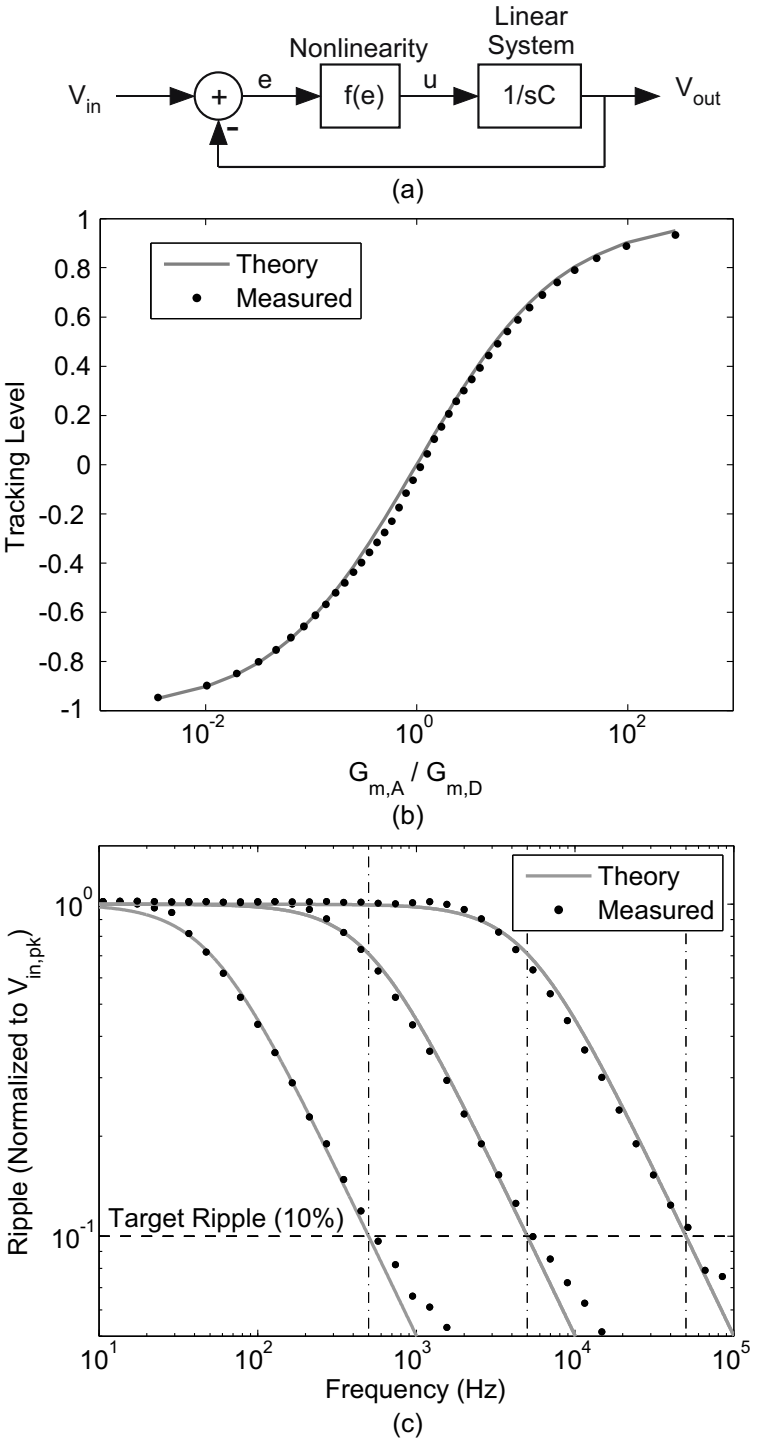

Figure 5.4: (a) Model that is used for analyzing the peak detector. (b) Peak detector tracking level as a function of the attack-to-decay ratio. (c) Ripple as a function of frequency for three different biases: targeting 10% ripple at 500Hz, 5kHz, and 50kHz. The data in (b) and (c) were measured with a peak detector fabricated on a 0.35μm CMOS process, since we did not have direct access to the peak detector output with the 0.18μm circuit.

$$R_e = \sqrt{1 + R_O^2 - R_O \cos(\phi)} \tag{5.3}$$

R_e is the amplitude at node e normalized by the input amplitude. As shown by (5.2), the tracking level depends on the ratio of transconductances and also on the output ripple; this was verified experimentally in Fig. 5.4(b), which shows how the tracking level varies with the attack-to-decay-ratio for a fixed output ripple ratio ($R_O = 3\%$). As expected, the tracking level is zero/centered when $G_{m,A} = G_{m,D}$, and the tracking level increases as $G_{m,A}$ is increased above $G_{m,D}$, saturating as the tracking level approaches 100%.

By solving the loop at the fundamental frequency, ω, the following equation was obtained for the output ripple, R_O

$$\frac{\omega C_{PD} R_O}{G_{m,D} R_e} = \frac{R_g + 1}{2} + \frac{1 - R_g}{\pi} \sin^{-1}\left(\frac{A_t}{R_e}\right) + \frac{1 - R_g}{\pi} \frac{A_t}{R_e} \sqrt{1 - \frac{A_t^2}{R_e^2}} \tag{5.4}$$

where $R_g = G_{m,A}/G_{m,D}$. In normal operation, the peak detector is tuned for a particular operating frequency by using the transconductances to obtain the desired ripple at that frequency. This procedure is demonstrated in Fig. 5.4(c), wherein (5.2) and (5.4) are used to bias the circuit for $R_O = 0.1$ (10% ripple) at three different frequencies: 500Hz, 5kHz, and 50kHz. The ripple has a first-order lowpass dependence on frequency, since the circuit is a first-order asymmetric integrator. The ripple increases as the frequency decreases, until the frequency is below the corner frequency of the peak detector, at which point the peak detector acts as a follower. In order to maintain peak detector operation, the signal frequency must remain within the −20dB/decade slope region; if the signal frequency drops too low, then the circuit no longer performs rectification. Thus, if the circuit is going to be used for broadband operation, it should be biased such that the lowest frequencies of interest remain above the corner frequency. For example, in the broadband speech demonstration of Fig. 5.11, the peak detector was biased for 30% ripple at 200Hz, which is within the range of fundamental frequencies for speech [109]. If the circuit is used for subband operation, such as the filter bank in Fig. 5.1(b), then each band's detector is biased to have the desired ripple at that subband frequency.

5.3.3 Peak Detector Biasing

The following is a procedure for using (5.2) and (5.4) to choose the attack and decay rates required to operate the circuit at a specified tracking level, ripple level, and input frequency.

1. Specify the tracking level A_t, ripple level R_O, and operating frequency ω. If biasing for envelope-detector operation (i.e. tracking to the top of each peak), use $A_t = 1 - R_O$
2. Initialize $\phi = -\pi/2$
3. Use (5.2) to solve for the attack-to-decay ratio $G_{m,A}/G_{m,D}$
4. Use (5.4) to solve for the decay rate $G_{m,D}$
5. Refine $\phi = \tan^{-1}\left(-\frac{\omega C_{PD}}{G_{m,D}}\right)$ and repeat steps 3-4

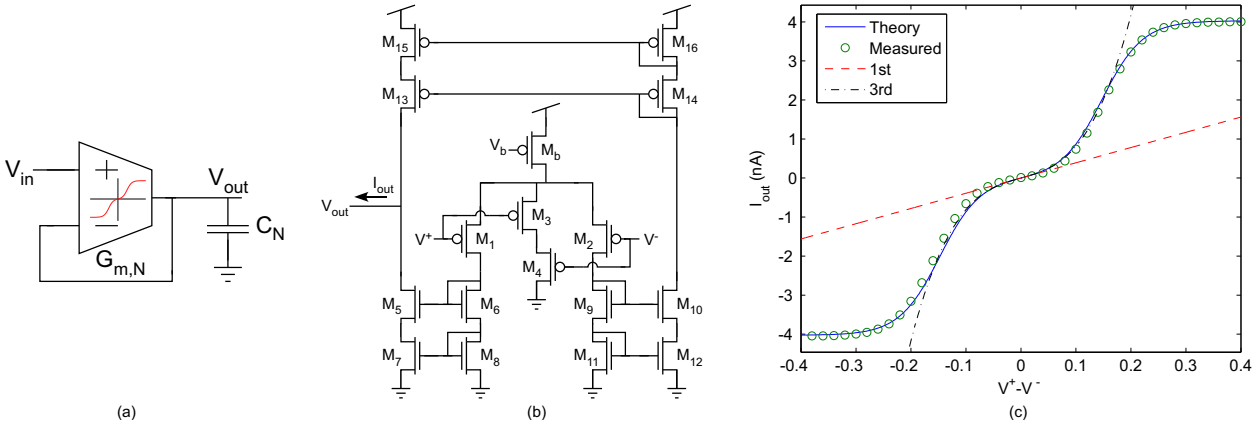

Figure 5.5: (a) The adaptive-τ filter is a follower-integrator where the transconductance element has an expansive nonlinearity. (b) The nonlinear transconductance element is a pFET-based OTA with bump de-linearization. (c) I–V curve for the nonlinear OTA, shown with first- and third-order Taylor series expansions.

6. (Optional) To bias for envelope-detector operation, use $G_{m,A}/C_{PD} = 10\omega$ to ensure the output reaches the peaks with aligned phase

5.4 Adaptive-Time-Constant Filter

As discussed in the beginning of the chapter, the second stage of the magnitude detector integrates the first stage's initial magnitude estimate, removing the ripple that couples in from the carrier signal to produce a smooth magnitude estimate. To obtain a response with low ripple and high temporal accuracy, we have developed a lowpass filter with an expansive nonlinearity to achieve an amplitude-dependent time constant. We call this filter an adaptive-time-constant filter, or adaptive-τ filter. The expansive nonlinearity is achieved by using a nonlinear transconductance element in a follower-integrator filter topology, as shown in Fig. 5.5(a). The transconductance element has a *sinh*-shaped voltage-to-current relationship. Thus, for small differential voltages, it has a low and essentially linear transconductance, resulting in a long time constant for suppressing ripple; for large differential voltages, the transconductance increases, resulting in a shorter time constant to provide a better temporal response. In this Section, we present this adaptive-τ filter.

5.4.1 Nonlinear Transconductor

We have formed the expansive nonlinearity by using a "bump circuit" within a standard OTA (see Fig. 5.5(b)) [121]. Such "bump-OTAs" have been used to create linearized transconductors through appropriate sizing of the "bump" transistors, M_3 and M_4 [98, 99]. Here, we have used the bump transistors to design the cubic nonlinearity in Fig. 5.5(c) that is used in our adaptive-time-constant filter; a similar nonlinear transconductor was used for circuits implementing Hebbian learning [122]. In the bump-OTA, the current through the

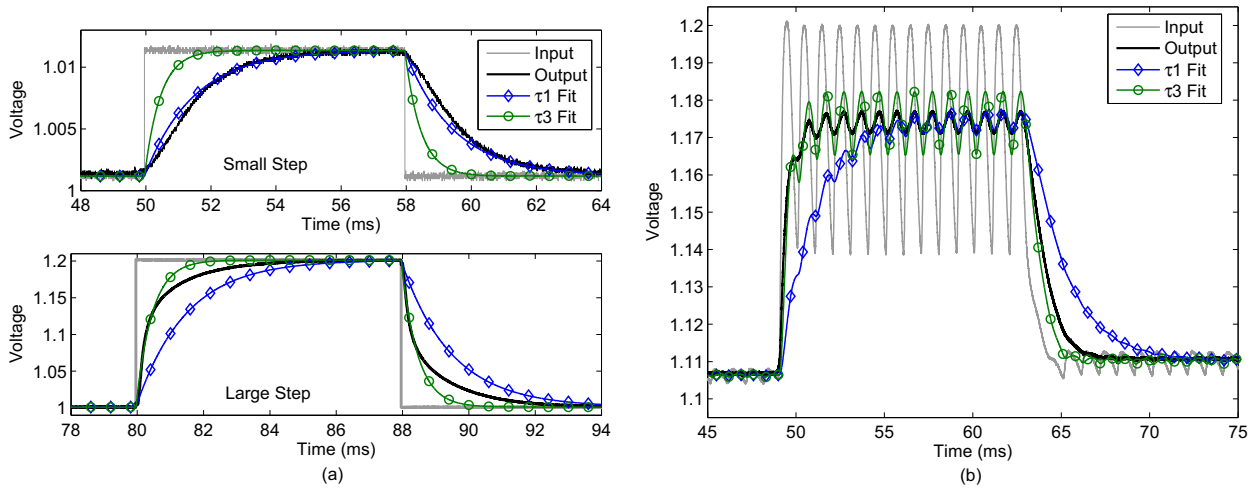

Figure 5.6: (a) Measured step response of the adaptive-τ filter, shown with simulated first-order linear filters which correspond to the adaptive-τ filter's effective time constants for small and large amplitudes. (b) Measured output of the adaptive-τ filter in response to the peak detector's output, shown with the same simulated first-order linear filters as in (a).

tail transistor (M_b) is shared by the input transistors (M_1 and M_2) and the bump transistors (M_3 and M_4). The current through the bump transistors is greatest when $V^+ = V^-$. By making the bump transistors have a large $\frac{W}{L}$ ratio, they steal a significant amount of current from the input pair, creating a low-transconductance region for small differential voltages and generating the expansive nonlinearity. The voltage-current relationship for this circuit is described by

$$I_{out} = I_b \frac{\sinh\left(\frac{\kappa}{U_T}\left(V^+ - V^-\right)\right)}{1 + S/2 + \cosh\left(\frac{\kappa}{U_T}\left(V^+ - V^-\right)\right)} \tag{5.5}$$

where κ is the subthreshold slope, U_T is the thermal voltage, and the strength parameter $S = (\frac{W}{L})_{3,4}/(\frac{W}{L})_{1,2}$ is the relation between the aspect ratio of the bump transistors and the input pair [98]. The voltage-to-current relationship for our OTA is shown in Fig. 5.5(c), and the first two nonzero Taylor series coefficients are

$$a_1 = I_b \frac{\kappa}{U_T} \frac{1}{2 + S/2} \qquad a_3 = I_b \left(\frac{\kappa}{U_T}\right)^3 \frac{S/2 - 1}{6(S/2 + 2)^2} \tag{5.6}$$

These coefficients are shown with the V-I sweep in Fig. 5.5(c). For small differential voltages, the nonlinear-OTA acts as a linear transconductor with transconductance a_1. Increasing the input-output differential voltage increases the effective transconductance according to a_3.

5.4.2 Demonstration of Performance

Figure 5.6(a) demonstrates the step response of the adaptive-τ filter. The response is shown for two steps of different sizes: a small step for which the linear term dominates

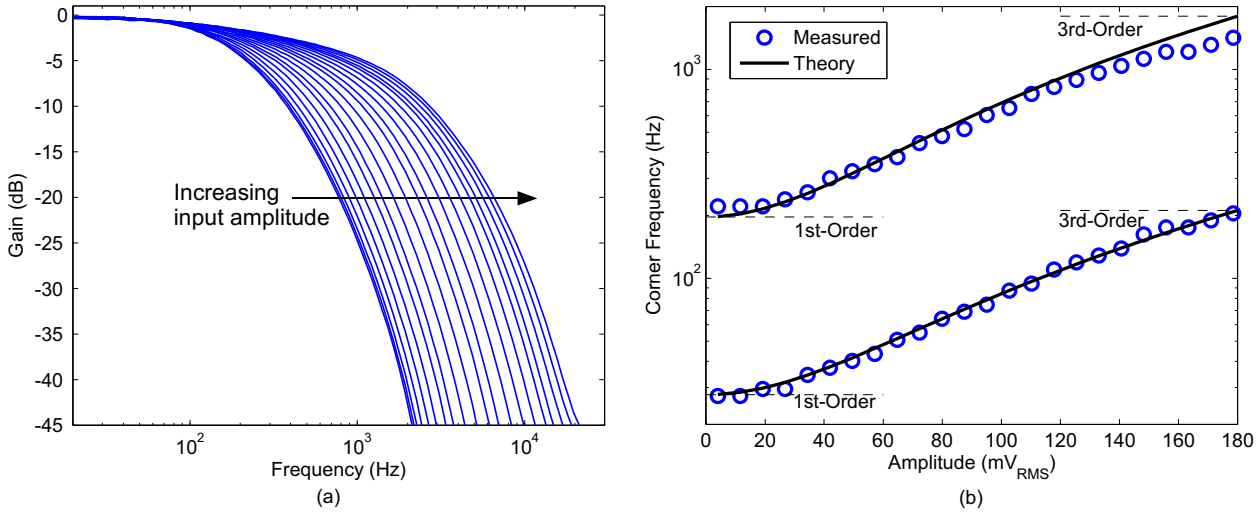

Figure 5.7: (a) Measured frequency response of the adaptive-τ filter. Each line is a frequency response for a different input amplitude. (b) Corner frequency of the adaptive-τ filter as a function of the input amplitude, shown for two different OTA biases.

(top pane) and a large step for which the higher-order terms dominate (bottom pane). Shown with the measured response of the adaptive-τ filter are the responses of two simulated first-order *linear* filters: the τ_1 filter has a time constant corresponding to the linearized transconductance for small signals (i.e. $\tau_1 = C_N/a_1$), and the τ_3 filter uses a shorter time constant corresponding to the linearized transconductance for the large step (i.e. the adapted time constant, $\tau_3 = C_N/(a_1 + a_3\frac{3}{4}V_{step}^2)$). For the small step, the adaptive-τ filter's response follows the τ_1 filter's response since the first-order term dominates. For the large step, the adaptive-τ filter initially follows the τ_3 filter's response, but it reverts to the longer time constant of the τ_1 filter as it gets close to the final value of the step. This changing time constant helps the filter achieve a faster response for large transients.

To motivate the choice of the adaptive-τ filter for the magnitude detector, we exhibit the experiment in Fig. 5.6(b), which compares the performance of the adaptive-τ filter with the two simulated first-order linear filters used in the experiment of Fig. 5.6(a). The input to the filters is the response of the peak detector to a sine wave stepped from 5mV_{pk} to 100mV_{pk} and then to 10mV_{pk}. The τ_1 filter yields the same ripple as the adaptive-τ filter but cannot follow the steps closely in time. The τ_3 filter follows the steps but has more ripple than the adaptive-τ filter. These results show that the adaptive-τ filter achieves a good tradeoff between ripple suppression and temporal response, while also being compact and low-power.

5.4.3 Design

To design the adaptive-τ filter, we need to know how its time constant depends on the input amplitude, bias current, and strength parameter (S). Here we develop an approximation to relate those parameters and then show how to use this approximation to design and bias the circuit. Our approximation is based on the describing function [119]

of the nonlinear transconductance element. The filter has the form of Fig. 5.4(a) with $f(V_{in} - V_{out}) = a_1 (V_{in} - V_{out}) + a_3 (V_{in} - V_{out})^3$, where a_1 and a_3 are the Taylor series coefficients given by (5.6) and are controlled by the bias current and the strength parameter. The sinusoidal-input-describing-function (i.e. an amplitude-dependent gain term) for this nonlinearity is $a_1 + a_3 \frac{3}{4} |V_{in} - V_{out}|^2$ [123]. Knowing that the input-output differential is related to the input amplitude $V_{in,pk}$, the transfer function can be approximated as

$$H(s, V_{in,pk}) \approx \frac{1}{1 + sC_N/(a_1 + a_3 \frac{3}{4} V_{in,pk}^2)} \tag{5.7}$$

where the corner frequency has a quadratic relation to the input amplitude. Equation (5.7) gives an approximate transfer function for the circuit and is verified with real data in Fig. 5.7. Figure 5.7(a) shows the frequency response measured at different input amplitudes and demonstrates an increasing corner frequency for increasing amplitude. Figure 5.7(b) shows the variation in corner frequency as a function of input amplitude for two different filter biases. The circles show the measured corner frequencies and the solid lines show the predicted values using $2\pi f_c = (a_1 + a_3 \frac{3}{4} V_{in,pk}^2)/C_N$ (i.e. the corner frequency in (5.7)). This experiment verifies that the corner frequency is a function of the square of the input amplitude.

Using (5.7) and the definitions of a_1 and a_3, the circuit can be designed to exhibit the desired ripple suppression and transient response. The procedure to design and bias the filter is:

1. Specify (a) the ripple-suppression time constant, τ_{rip}, (b) the transient-response time constant, τ_{tran}, and (c) the amplitude that is considered a transient (and should be followed with the transient-response time constant τ_{tran}), A_{tran}
2. Use the ripple-suppression time constant to compute $a_1 = \frac{C_N}{\tau_{rip}}$
3. Use a_1 to find the value of a_3 that yields the desired transient-response time constant at an amplitude of A_{tran} by using $a_3 = \left(\frac{C_N}{\tau_{tran}} - a_1 \right) \frac{4}{3A_{tran}^2}$
4. Use (5.6) to find S in terms of a_1 and a_3 by using $S = 2 \frac{(\kappa/U_T)^2 + 12 a_3/a_1}{(\kappa/U_T)^2 - 6 a_3/a_1}$
5. Use (5.6) to find the bias current $I_b = a_1 \frac{U_T}{\kappa} (2 + S/2)$

5.5 Complete Magnitude Circuit

In this Section, we present the complete magnitude detector circuit, shown schematically in Fig. 5.1(a). This circuit is formed by combining the peak detector of Section 5.3 with the adaptive-τ filter of Section 5.4. Figure 5.8 shows a micrograph of the circuit, which was fabricated in a standard 0.18μm CMOS process. The values used for capacitors C_{PD} and C_N are 4.5pF and 3pF, respectively. The total area of the circuit is 0.019mm^2.

As discussed in Section 5.2, our objective is to develop a magnitude detector with improved temporal accuracy, while maintaining a low-ripple response. This tradeoff between

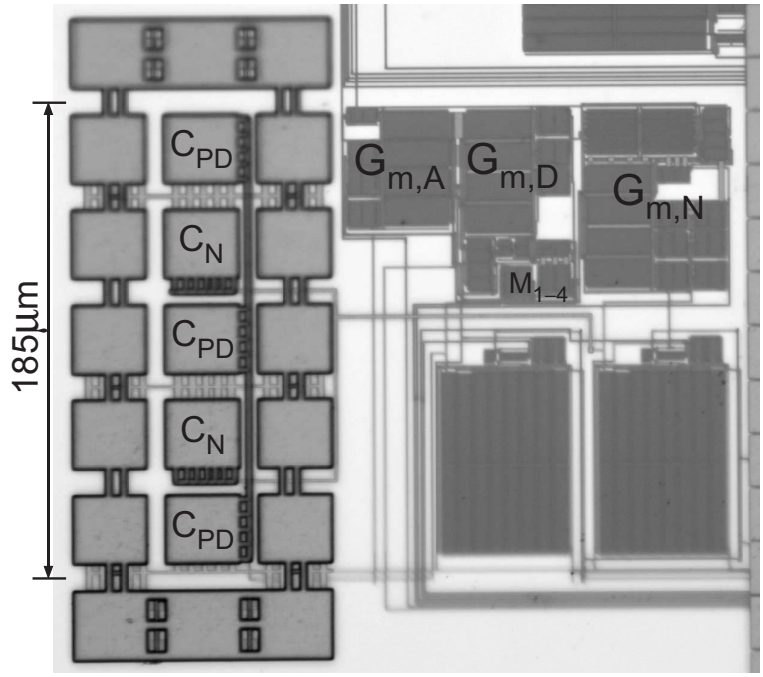

Figure 5.8: Micrograph of the magnitude circuit, which was fabricated in a standard 0.18μm CMOS process.

Table 5.1: Tradeoff Between Ripple and Acquisition Time

Magnitude Structure	$\omega\tau_{aq}$	$\omega\tau_{aq}$ Improvement (1% Ripple)
PD Only	$1/R_T$	1
PD w/ LPF	$\sqrt{2/R_T}$	7.1
PD w/ 2nd-Order LPF	$\sqrt{3}/\sqrt[3]{R_T}$	11.3
PD w/ Adaptive-τ	$\sqrt{\frac{1}{4R_T}+\frac{4}{R_T\left(1+A_s^2\frac{3a_3}{4a_1}\right)^2}}$	17.7

ripple and temporal accuracy can be understood graphically from Fig. 5.4(c). For example, a standalone peak detector can be tuned for low ripple by biasing the circuit's corner frequency, ω_c, to be much less than the operational frequency, ω, which is the frequency for the desired target ripple. This biasing results in a long acquire time since the acquire time constant, τ_{aq}, is given by $\tau_{aq} = 1/\omega_c$ and is long compared to the input signal's period at the operational frequency.

The $\omega\tau_{aq}$ product is related to the number of cycles required to acquire transients and is thus a good metric for temporal responsiveness. In Table 5.1, we compare $\omega\tau_{aq}$ for different magnitude-detection architectures for a given total ripple at the output, R_T. The third column states the factor of improvement for each architecture with 1% total ripple over the

baseline case of the standalone peak detector (PD). For example, a PD combined with a lowpass filter (LPF) is 7.1 times faster than a PD by itself. This improvement is because adding an LPF to the PD increases the slope of the ripple-frequency relationship (i.e. the slope of Fig. 5.4(c)), thereby moving ω_c closer to ω; the best result is obtained by splitting the ripple suppression evenly between the PD and the linear LPF (e.g. 10% suppression in each stage to obtain 1% total suppression). Using a second-order filter further improves the temporal accuracy by increasing the slope of the ripple-frequency relationship. However, this technique only achieves 1.6-times improvement beyond the first-order filter, while adding significant increases in power, area, and frequency sensitivity (due to the larger slope). With the adaptive-τ filter, we operate the peak detector faster than with the linear filter (typically twice as fast) which, accordingly, yields more ripple at the PD's output. To achieve the same total ripple at the output of the magnitude detector, the value of a_1/C_N in the adaptive-τ filter is tuned to compensate for the increased PD speed. Large transients then cause the filter's time constant to decrease such that the PD's speed is the main limitation to the magnitude detector's overall speed. As can be seen by the Table, the adaptive-τ filter yields a 2.5-times improvement in temporal response for an amplitude step (A_s) of 300mV as compared to the linear filter and without the cost of the second-order filter.

Furthermore, we have compared the acquisition times—defined as the number of cycles to reach 99% of a step—of the adaptive-τ ripple suppression case with the first-order linear filtering case, for steps between the minimum ($200\mu V_{RMS}$) and maximum ($630mV_{RMS}$) detectable signals of this circuit. For an upward step, the acqusition time improves from 7.58 cycles with a linear filter to 1.45 cycles with the adaptive-τ filter; and for a downward step, the acquisition time improves from 18.3 cycles with a linear filter to 14.1 cycles with the adaptive-τ filter.

Figure 5.9 shows the measured dynamic range of the magnitude circuit for operational frequencies of 200Hz and 2kHz (i.e. typical speech frequencies). The biasing routines discussed in Sections 5.3.3 and 5.4.3 were used to bias the circuit to track the RMS with 1% ripple. The response remains within 1dB linearity from $200\mu V_{RMS}$ to $630mV_{RMS}$ for both cases, yielding a dynamic range of 70dB. These two measurements are characteristic of the performance of this circuit for typical speech frequencies, including those covering the telephony frequency range (i.e. $\leq$5kHz). Additionally, the dynamic range was measured to be 64dB at the upper end of the audio frequency range (20kHz).

The minimum detectable signal is limited by the greater of two nonidealities: the noise or the "deadzone" (which is created by offsets in the peak detector's OTAs). Since the magnitude detector can easily be designed to achieve a noise floor below typical offsets for this type of OTA architecture, the designer should focus on the offsets. For example, our complete magnitude detector was measured to have $\leq 115\mu V_{RMS}$ for all biasings across the audio frequency range, which is well below the minimum detectable signal of $200\mu V_{RMS}$, indicating that offsets dominated our minimum detectable signal. Offsets in the peak detector OTAs affect overall circuit operation as follows. Due to the symmetry of the peak detector, its minimum detectable signal is not degraded when $G_{m,A}$ and $G_{m,D}$ have equal offsets with the same sign. The minimum detectable signal is only degraded by a mismatch between the offsets, which will either create a gap between the attack and decay states or will cause an overlap of the attack and decay states; both cases compromise the accuracy of the magnitude estimate for signals smaller than the difference between the offsets. Thus, the most critical

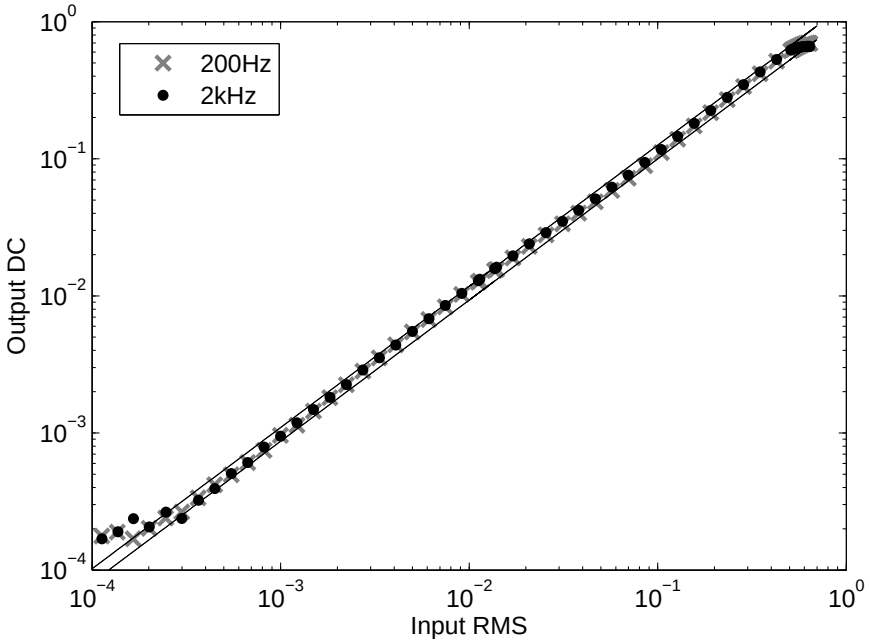

Figure 5.9: Dynamic range measurement of the complete magnitude circuit. The response remains linear to within 1dB across a range of 70dB.

factors for improving the minimum detectable signal are minimizing the offsets and matching the peak detector OTAs to ensure similar offsets with low variance.

To verify the simultaneous achievement of temporal and amplitude accuracy, we performed the experiments of Figs. 5.10–5.11. In both experiments, the circuit was biased for RMS tracking and 1% ripple. In Fig. 5.10, a sine wave with a frequency of 2kHz had its amplitude modulated by six Gaussian pulses with amplitudes increasing logarithmically from 2mV_{pk} to 500mV_{pk}. The first three pulses are shown in the left panes and the last three pulses are shown in the right panes with different y-axis limits. The top panes show the peak detector response and the bottom panes show the magnitude response along with the actual RMS. Figure 5.11 shows the response of the magnitude circuit to a speech signal. A mathematically calculated RMS is shown alongside the circuit's response. In both Figs. 5.10 and 5.11, we see that the magnitude circuit closely follows the same shape as the RMS calculations, and that the magnitude detector accurately follows the quickly changing RMS across a wide dynamic range.

The characteristics of this circuit are summarized in Table 5.2 and are compared with relevant state-of-the-art magnitude detectors. The power consumption of our circuit scales linearly with the operational frequency (since transconductance scales linearly with current in subthreshold operation); for typical biasing, the power varies from 1.1nW–1.08μW across the 20Hz–20kHz audio frequency range. The figure of merit (FOM) in the table is defined as $FOM = 10^{DR/20} f_{max}/P$, where DR is the dynamic range, f_{max} is the maximum operational frequency, and P is the power consumption. Our FOM number is given for a frequency of 200Hz, since that is the tuning condition used to operate on wideband speech signals.

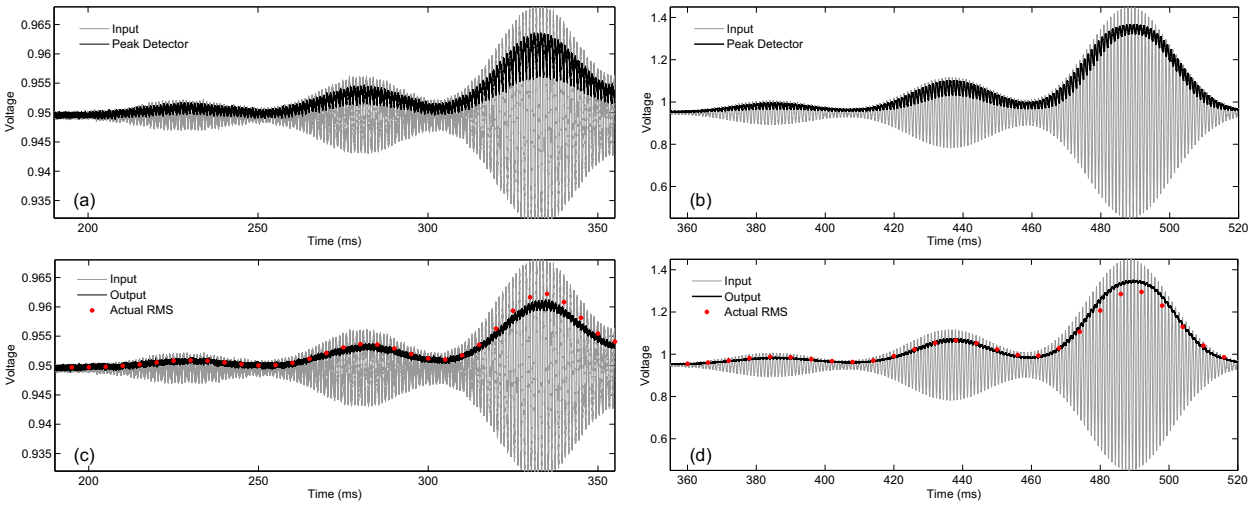

Figure 5.10: Measured transient response of the magnitude detector for logarithmically increasing amplitude from 2mV_{pk} to 500mV_{pk}.

Table 5.2: Magnitude Detector Comparison

	Process	V_{dd}	Freq. Range	Dyn. Range	Power	FOM	Comments
Proposed	0.18μm	1.8V	20Hz–20kHz	70dB (200Hz) 70dB (2kHz) 64dB (20kHz)	1.1nW–1.08μW	57.9	DR <1dB nonlin.
[124]	1.5μm	2.8V	100Hz–10kHz	75dB	2.8μW	20.1	
[125]	1μm	3V	12MHz	54dB	10mW	0.60	
[126]	0.18μm	1.8V	100Hz–1.6GHz	50dB	6.3mW	80.3	
[127]	0.35μm	1V 1.5V	100Hz	73dB 83dB	60nW 90nW	7.44 15.7	Rectifier only
[128]	0.35μm		1–10MHz	42dB	2.98mW	0.42	Simulated; DR <1dB nonlin.

5.6 Conclusion

In this work, we have presented a low-power magnitude detector circuit, which achieves good temporal responsiveness through the use of a novel peak-detector-nonlinear-integrator topology. The circuit was built in a 0.18μm CMOS process. At 200Hz, which is a typical operating point for wideband speech signals [109], the circuit achieves a dynamic range of 70dB with a power consumption of 10.92nW. The compactness and low-power operation of the circuit, combined with its flexible biasing, make it a good choice for applications such as spectral analysis.

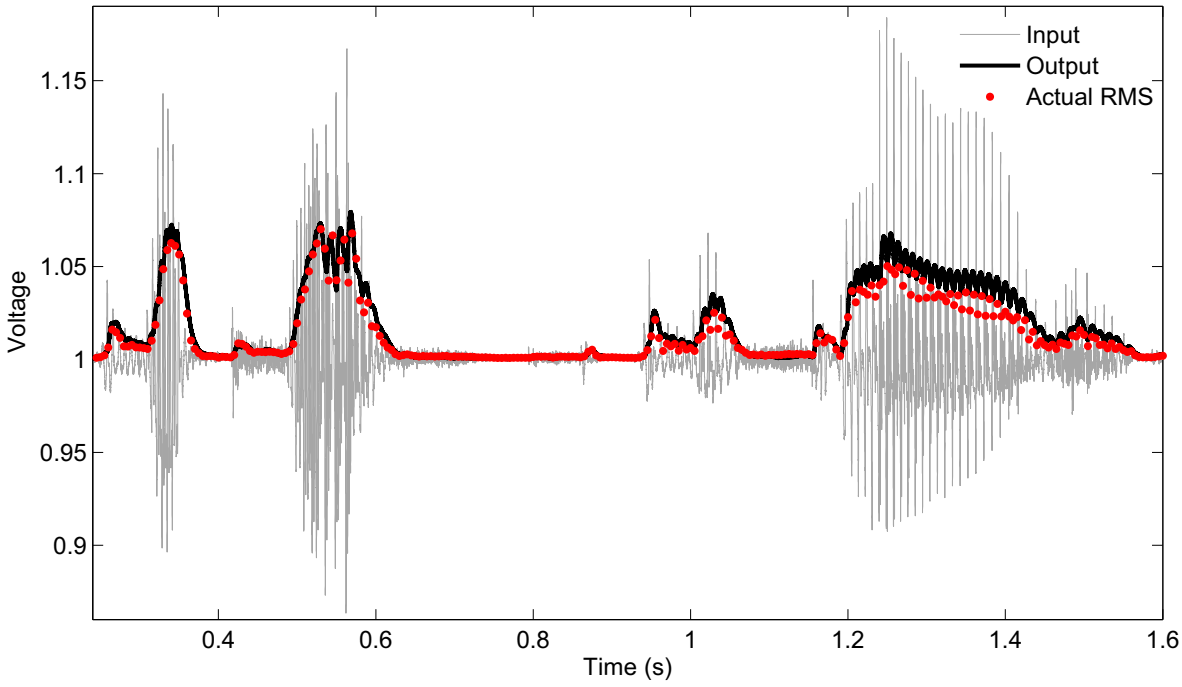

Figure 5.11: Measured response of the magnitude circuit to a speech waveform. For comparison, the response is shown with a computer calculated RMS.

Chapter 6

Floating-Gate Transistors for Programmable Analog Circuitry

In Chapter 3, we examined the benefits that an analog front-end can provide in a low-power sensing application. The primary shortcoming of that front-end was the method of controlling the circuit parameters (e.g. filter frequencies). We used a resistive divider for parameter biasing, which prevented independent biasing of channel parameters, limited the parameter biasing resolution to 2.5mV (≈7% deviation in center frequency), and worst of all, the parameter biasing accounted for over 70% of the quiescent power consumption. To design a flexible and efficient analog front-end for wireless sensor networks, we need programmable analog parameter storage that is dense and low power. Floating-gate transistors are an excellent option for analog storage. They have been used for assorted applications [83, 97, 129, 130], and we have used them for our programmable filter bank in Chapter 4, for our new analog front-end in Chapter 9, and for our field-programmable analog array in Chapter 10.

As an apt choice for long-term analog memory in standard CMOS processes, floating-gate transistors are key enablers for large-scale programmable analog systems. Such systems are often designed for battery-powered—and generally resource-constrained—applications, which require the memory cells to program quickly with low infrastructural overhead. In order to meet these needs, we present a new analog floating-gate memory cell in this Chapter. Our four-transistor memory cell offers both voltage and current outputs and has linear injection and tunneling characteristics. Furthermore, we present a simple programming circuit that forces the memory cell to converge to voltage targets within 100ms and with 8-bit accuracy.

The work in this Chapter was published in the Proceedings of the International Midwest Symposium on Circuits and Systems [131]. Furthermore, these developments have been incorporated into our FPAA in Chapter 10.

6.1 Floating-Gate Transistors

In addition to their role as nonvolatile memory elements in Flash memory, floating-gate (FG) transistors are used for programmable voltage/current references, precision analog device matching, and adaptive/learning circuits [132]. An FG transistor [schematic repre-

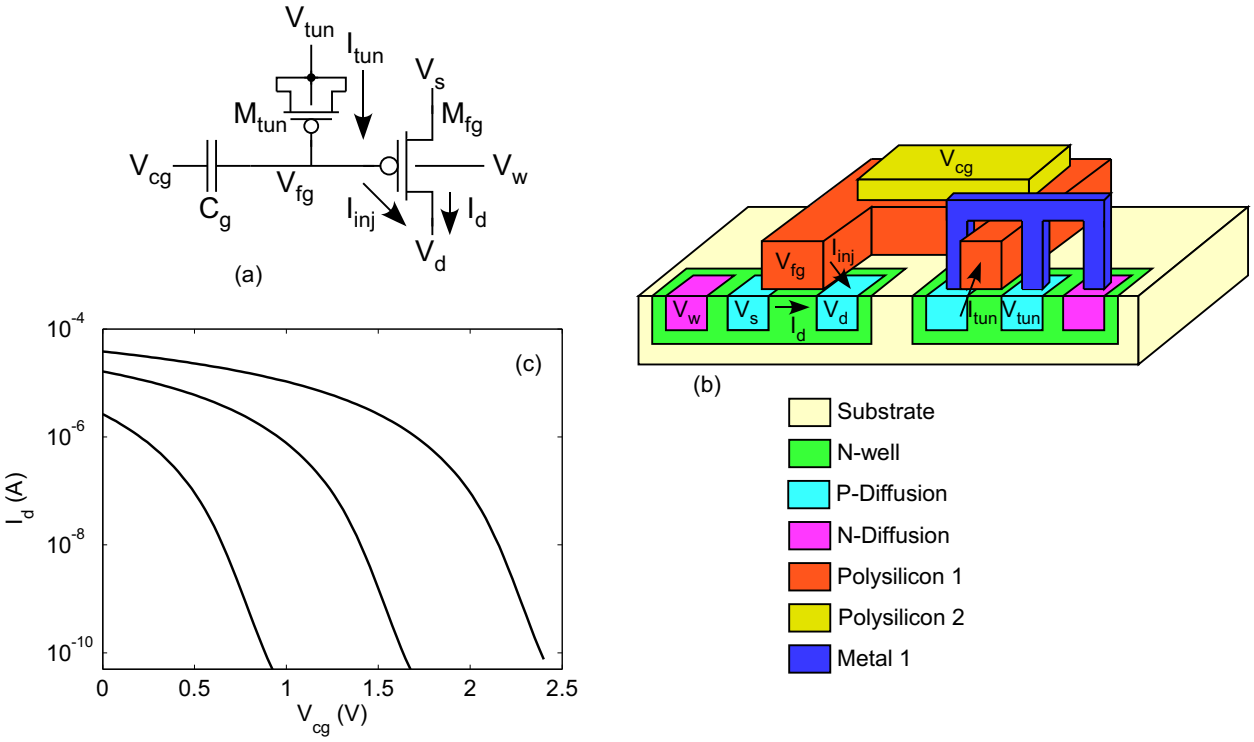

Figure 6.1: Overview of floating-gate (FG) transistors. (a) Circuit symbol. (b) 3-D layout of an FG transistor. The first polysilicon layer is used for the floating gate, and the second polysilicon layer is used for the control gate. Electrons are injected onto the FG from from the drain and are tunneled from the FG to V_{tun}. (c) Measured gate sweeps of an FG transistor. The sweep was performed three times for three different programmed values in order to demonstrate the ability to shift the threshold voltage.

sentation shown in Fig. 6.1(a)] is a MOSFET (M_{fg}) that has no resistive connection to its gate; instead, a "control gate" (V_{cg}) couples capacitively onto the transistor's "floating gate" (V_{fg}). As a result, the FG's charge, which can be modified using Fowler-Nordheim tunneling and hot-electron injection, creates a programmable and nonvolatile threshold-voltage shift from the perspective of the control gate. Figure 6.1(c) shows sweeps of the control-gate voltage for three different levels of programmed charge, thus illustrating the programmable shift in threshold voltage.

Figure 6.1(b) shows a three-dimensional view of an FG transistor in a standard CMOS process. The floating gate is the first layer of polysilicon, and the second layer of polysilicon is used for the control gate. To reduce the size and layers, the control gate may be formed using a MOS capacitor, but the linear characteristics of poly-insulator-poly capacitors simplifies the incorporation of FGs into analog circuitry.

As in a standard MOSFET, the drain current of an FG transistor is determined primarily by the voltage difference between the source and the polysilicon gate over top of the channel. The unique characteristics of an FG transistor result from the fact that the dc gate voltage, $V_{\rm fg}$, is determined simultaneously by the charge on the gate and by the voltages which

couple through capacitive division onto the gate. If the charge on the FG is Q and the total capacitance on the FG is C_T, then

$$V_{fg} = \frac{Q + C_g V_{cg} + C_d V_d + C_s V_s + C_{tun} V_{tun} + C_w V_w}{C_T} \tag{6.1}$$

where C_g, C_d, C_s, C_{tun}, and C_w are the capacitances that couple from V_{fg} to V_{cg}, V_d, V_s, V_{tun}, and V_w, respectively. Typically, the control-gate capacitance C_g is designed to dominate the coupling terms and most of the voltages are constant. So a simpler equation

$$V_{fg} = \frac{Q + C_g V_{cg}}{C_T} \tag{6.2}$$

is often sufficient.

Once the expression for V_{fg} is known, then the operating region and drain current of M_{fg} can be determined in the same manner as for any MOSFET. Using (6.1), the current for a MOSFET in the subthreshold region[1] is

$$I_d = I_0 \frac{W}{L} \exp\left(\frac{\kappa\left(Q + C_g V_{cg} + C_d V_d + C_s V_s + C_{tun} V_{tun}\right)}{C_T U_T}\right)\left[\exp\left(-\frac{V_s}{U_T}\right) - \exp\left(-\frac{V_d}{U_T}\right)\right] \tag{6.3}$$

This equation illustrates that the difference between an FG transistor and a standard transistor is the addition of a linear transformation on the input voltage(s). This quality adds three unique capabilities.

1. The charge on the floating gate, Q, creates a programmable nonvolatile offset.
2. The superposition of multiple voltages onto the gate allows the use of multiple inputs to the transistor.
3. The scaling of input voltages via capacitive division allows manipulation of the relative weighting of voltages onto the gate.

6.2 Applications of Floating-Gate Transistors

The unique capabilities of FG transistors that were described in the previous section have led to various applications. The most common application is Flash memory wherein quantized data is stored as a charge on the gates [133]. FGs have also been adapted into Ion-Sensitive FETs (ISFETs), a type of chemical sensor that benefits from CMOS scaling [134]. Below, we describe the fundamental ways that FG transistors have been applied to analog circuit design.

6.2.1 Programmable Parameters

Most non-trivial analog circuits require the ability to vary circuit parameters. Examples include individual circuits such as programmable filters and programmable-gain amplifiers,

[1] See Appendix A for more on MOSFET modeling.

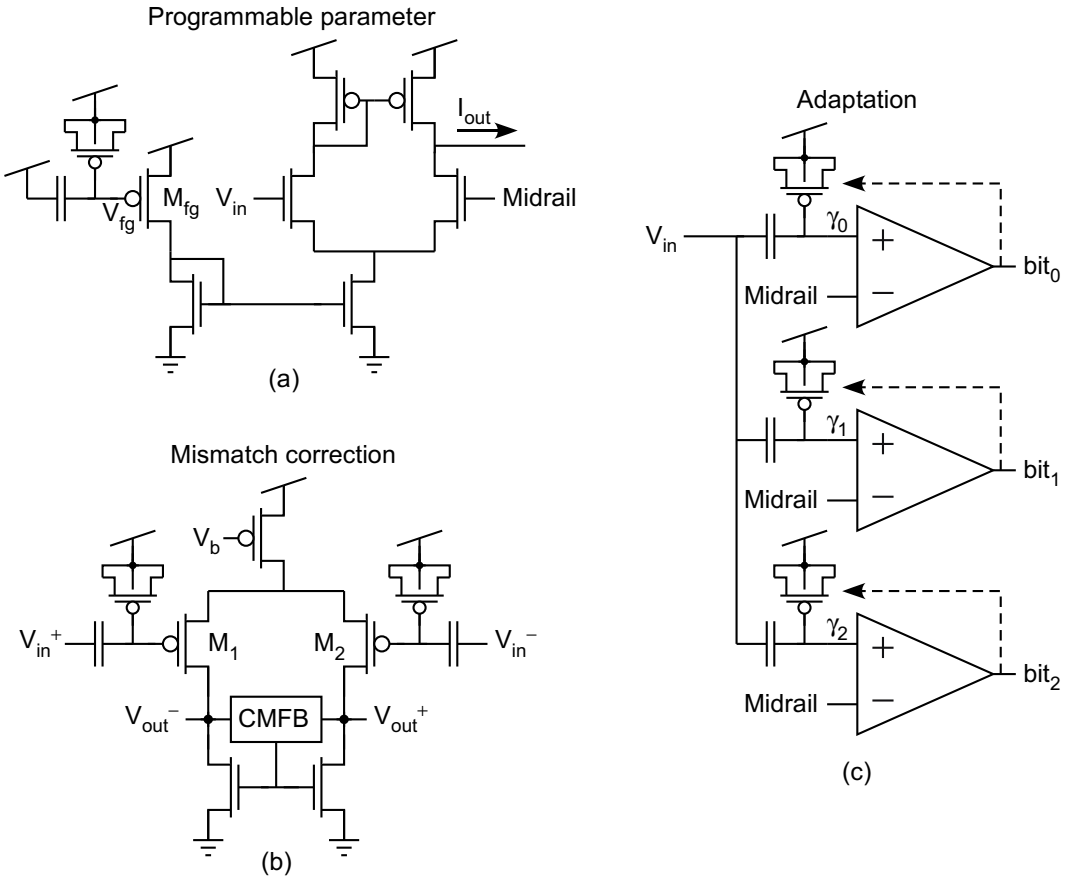

Figure 6.2: Applications of charge manipulation in floating-gate transistors. (a) M_{fg} used as a programmable current source to create a voltage-to-current converter with programmable transconductance. (b) Nulling the offset of an op-amp by using FG transistors for the input transistors. (c) Adaptation of quantization levels in a flash ADC as seen in [130].

as well as larger circuits and systems such as transceivers and field-programmable analog arrays. Traditional methods of creating programmable parameters have used digital-to-analog converters (DACs) or resistor/capacitor arrays. These methods use a large amount of chip area to obtain precise programming. As a result, these methods are inappropriate for systems with large numbers of parameters. FG transistors can provide programmable parameters by designing circuits with parameters that depend upon the charge on the floating-gate. Their small size makes FG transistors a good choice for programmable parameters. Figure 6.2(a) illustrates the use of an FG transistor in a programmable circuit. M_{fg} provides a programmable bias current to an OTA that is being used as a voltage-to-current converter. The OTA's transconductance, $G_m = f(V_{fg})$, is programmable since it is a function of the FG voltage, which is set by programming the charge. In this same manner, we have used FGs to create programmable reference voltages, reference currents, gains, time constants, and pulse lengths.

6.2.2 Precision Mismatch Correction

To achieve levels of precision that exceed the precision of device fabrication, all non-trivial analog circuits rely on the relative matching of devices, which is better than the absolute precision of the device parameters. Examples of circuit building blocks that rely on matching include current mirrors, differential pairs, and resistor/capacitor ratios. Nevertheless, device mismatch is one of the primary limitations in analog circuit performance. The use of FG transistors can improve precision because the threshold voltages of transistors that the designer wishes to match can be trimmed after fabrication. Figure 6.2(b) illustrates this concept. FG transistors are used for the input pair of an op-amp. By programming the relative charge on the two FGs, the threshold voltages of M_1 and M_2 can be adjusted to simultaneously cancel both the input pair mismatch and the NMOS current sink mismatch. This enables greater circuit precision.

6.2.3 Parameter Adaptation

Local parameter adaptation, such as is found in neural networks, allows efficient implementation of associative/perceptual processing applications. FG transistors are excellent for these applications since they combine memory storage and processing in a single device. Figure 6.2(c) illustrates how FG transistors can be used for parameter adaptation. In the Figure, FGs are used to adapt the quantization levels in a flash ADC [130]. The quantization level of each converter is stored as an FG charge. After each sample, the charge on an FG is incremented if the local *bit* is high, or decremented if the local *bit* is low. The quantization levels thus adapt over time to achieve a uniform distribution of output codewords, which maximizes the conversion entropy for a given signal. FGs have also been used for weight adaptation in neural networks.

6.2.4 Input Scaling

In the subthreshold region, the gate-to-source voltage increases by approximately 85mV for each decade of drain current.[2] It is sometimes desirable to scale this relationship, e.g. to increase the linear range of a differential pair. With an FG transistor, this input voltage can be scaled onto the floating gate using capacitive division. Then, the control-gate-to-source voltage will increase $\frac{C_g}{C_T}$85mV for each decade of drain current. Figure 6.3(a) illustrates an application of this concept. In the Figure, an OTA is used as a voltage-to-current converter similar to Fig. 6.2(a); the bias V_b may be controlled by a separate FG transistor. But in Fig. 6.3(a), an FG transistor (M_1) has been used to scale down the input voltage so that the linear range is increased by approximately $\frac{C_1+C_2}{C_1}$. Other methods for increasing the linear range of an OTA can achieve higher SNR when the amplifier's precision is noise-limited. But in wireless sensors that are small and low-cost, the amplifier's precision will typically be mismatch-limited because of the use of small transistors. In such applications, FG transistors can substantially increase the SNR by simulataneously increasing the linear range and reducing the mismatch (as described in Subsection 6.2.2).

[2]The precise relation depends upon the κ of the device.

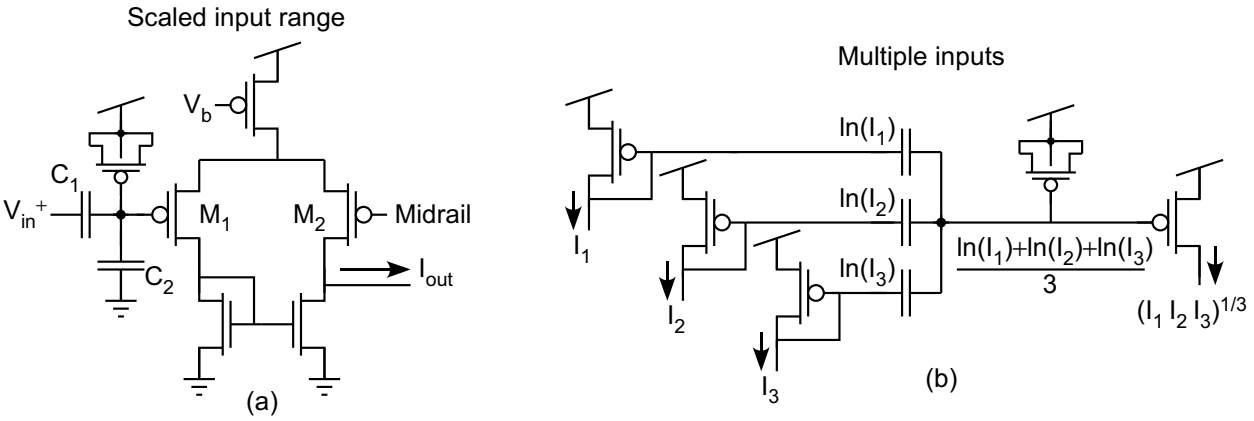

Figure 6.3: Applications of capacitive coupling in floating-gate transistors. (a) An extended range voltage-to-current converter achieved by scaling down the input with capacitive division. (b) A geometric mean function implemented using superposition of capacitively divided inputs to add and multiply in the log-domain.

6.2.5 Multiple-Input Transistors

Although addition operations are easy to construct in the current domain by simply summing currents on a node, voltage addition typically requires resistors or transconductors to perform voltage-to-current conversion prior to adding the currents. With floating-gate transistors, voltages can be summed onto the FG with multiple control-gate capacitors. Figure 6.3(b) illustrates this concept in a circuit that calculates the geometric mean of an array of currents. The circuit is essentially a current mirror with three inputs. The voltages from each input sum onto the floating gate. This sum is in the log-domain, so it will be transformed into a multiplication of the currents at the output. Additionally, the scaling from capacitor ratios divides each input by three. Again, this operation is performed in the log-domain, so at the output this scaling becomes a cube root. This class of circuits is known as MITE (multi-input translinear element) circuits and they are capable of compactly synthesizing a wide range of functions [135].

6.2.6 Summary of FG Transistor Applications

In this work, we have used FGs exclusively for programmable parameters. However, the research on FGs that is described in this Chapter and the subsequent Chapters—including continuous-time programming, charge manipulation characteristics, and high-voltage generation for charge manipulation—is relevant to all of these applications.

6.3 Overview of Floating-Gate Programming

Two phenomena are typically used to program FG transistors: hot-electron injection and Fowler-Nordheim tunneling.[3] Injection occurs when a large source-to-drain potential (V_{sd} > 3.5V for 0.35μm) is applied to the FG transistor, thus causing high-energy carriers to impact-ionize at the drain. A fraction of the resulting ionized electrons disperse toward the surface with enough energy to overcome the oxide barrier and inject onto the FG. In the subthreshold region, which is our target operational region, the injection current from V_{fg} to V_d can be approximated as

$$I_{inj} \approx \beta {I_s}^{\alpha} e^{V_{sd}/V_{inj}} \tag{6.4}$$

where I_s is the source current, and β, α, and V_{inj} are device-dependent fits [136]. Tunneling, on the other hand, requires high voltages (V_{ox} > 8V for 0.35μm). In order to avoid write disturbs during tunneling, unselected array elements must either be disconnected from the tunneling voltage using high-voltage switches or the FGs of the unselected elements must be raised to a sufficient voltage that tunneling does not occur [137]. Due to this difficulty in isolating tunneling within an array, tunneling is typically only used for global erasure in analog memory arrays, while injection is used to write to individual elements. Consequently, we focus mainly on injection in this work.

In order to modify the charge on the FG, high voltages are applied to the FG transistor's terminals. The charge can be programmed to a desired amount by using either pulsed or continuous methods, as illustrated in Fig. 6.4. Pulsed methods operate by applying short programming pulses and then measuring the FG after each pulse. In contrast, continuous methods continuously apply the programming voltage and use feedback to force the FG to converge to the target. Such continuous programming promises to be faster and require less peripheral circuitry than pulse-based programming.

Due to their ability to provide dense, low-power, analog biases, FGs are elemental in large-scale programmable analog systems—such as filter banks, classifiers, and field-programmable analog arrays. In these systems, circuit parameters (e.g. corner frequencies) are controlled by the charge on the FGs; as a result, system performance depends strongly on the programming accuracy. Prior pulse-based programming techniques have achieved high accuracy [105,138]. One advantage that pulse-based techniques have in terms of accuracy is that the FG is measured in a state that is similar to run-mode: with no high program voltages applied to the cell, and with the same current levels that will be used in run-mode. Unfortunately, pulsing is inherently slow due to the time spent reading, during which the high program voltages are stepped down and the FG is allowed to settle before the measurement is taken; if measuring low currents, then the read time further increases due to the long integration time that is necessary for accurate measurement. Methods to increase the programming speed rely on precise knowledge of each FG's characteristics, so that each pulse can move more aggressively towards the target [105]; but this adds to the complexity. Additionally, pulsing techniques require high-precision data conversion and pulse timing, and possibly large-range current measurement, all of which complicate the inclusion of analog FG memory into simple, resource-constrained, systems. Thus, there is a need for fast, compact, low-overhead, and accurate programming: we posit that continuous-time programming is more appropriate for

[3] These charge-manipulation mechanisms are described in detail in Chapter 7.

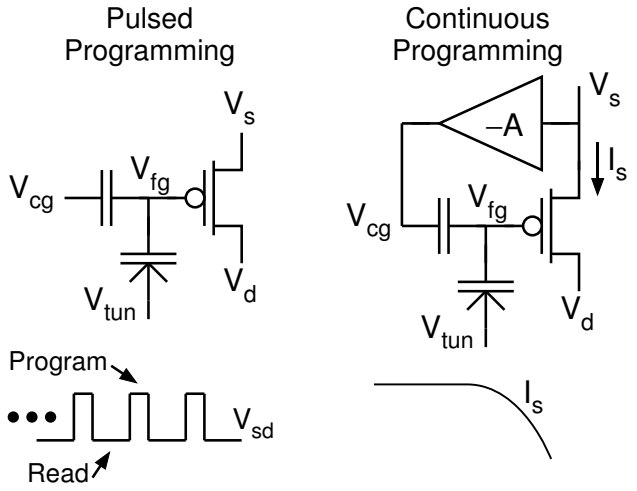

Figure 6.4: Pulsed programming and continuous programming. In pulsed programming, the source-to-drain potential is alternately pulsed high for injection, and then placed at a nominal value to measure the floating gate. In continuous programming, injection occurs constantly, and a terminal (in this case the source current) is adjusted to decrease—and eventually shut off—injection as the target is approached.

resource-constrained systems.

In the next Section, we describe our benchtop pulsed-programming system. This system was used as a sandbox for exploring programming dynamics, which led to the development of our integrated continuous-programming system that is presented later in the Chapter.

6.4 Pulse-Based Programming

Since FGs are used to create programmable circuit parameters (e.g. corner frequencies), system performance depends strongly on the programming accuracy. The standard technique for programming is the pulse-based technique, which is illustrated in Fig. 6.4(a). In this pulsed programming method, the source-to-drain voltage is alternately read at the nominal run-mode voltage and then pulsed to high programming voltages in order to inject charge.

We have developed a benchtop pulsed programmer for use within our lab. This programmer has been used to perform the FG programming in Chapters 4 and 9, but has since been phased out in favor of the low-overhead programming method in Section 6.6. A block diagram of the FG programmer is shown in Fig. 6.5. The programmer is controlled using Matlab, and as a result, programming algorithms can be quickly prototyped. The data-acquisition (DAQ) card in the setup is the PCI-6259. One digital output is used to switch tunneling on or off. Three analog outputs are used to control V_{cg}, V_d, and the FG transistor's V_{dd}. These outputs are streamed out synchronously at 300kHz, thus allowing near-microsecond resolution of pulse characteristics. The picoammeter is the Keithley 6485, which can perform up to 1000 readings per second and is interfaced to a PC using a GPIB cable. Below we discuss the two programming modes: coarse programming and fine programming.

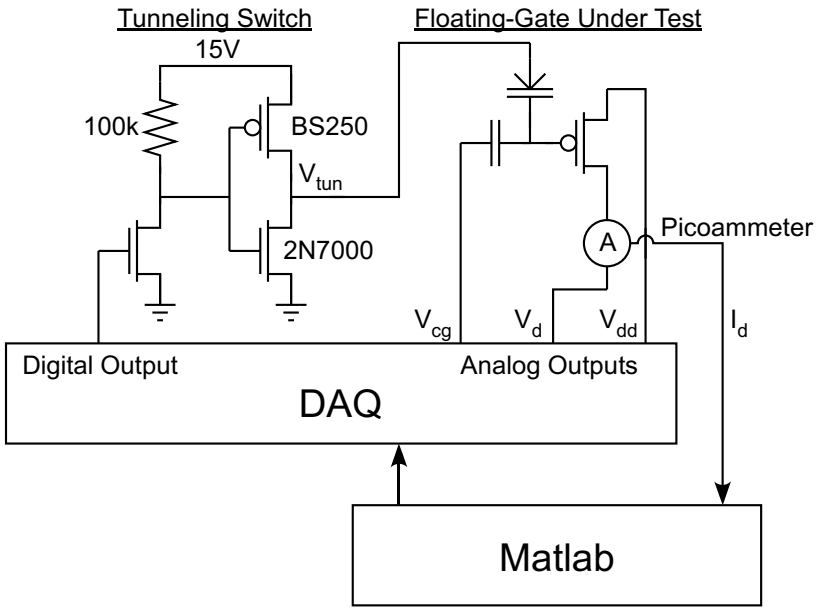

Figure 6.5: Block diagram of our benchtop floating-gate programmer.

6.4.1 Coarse Programming Mode

The coarse programming method is used to quickly inject the FG to the desired range. This is accomplished by continuously injecting rather than pulsing. The reason for continuously injecting is as follows. Although the picoammeter can perform up to 1000 readings per second, pulsing requires the picoammeter to switch to a high-current measurement range for each pulse so that the ammeter's voltage burden does not limit the source-to-drain voltage (and thus limit injection). This switching can cause up to one second of overhead for each pulse depending on the current level. In contrast, continuous injection allows a constant drain current and thus does not require any time-consuming ammeter range changes.

The coarse programming method has two feedback loops: one to linearize the injection characteristics and the other to make the programmer converge to a target. The control portions of these loops are shown in Fig. 6.6.

The injection linearization loop works by adjusting V_{cg} in order to maintain a constant I_d and thus to achieve a linear injection rate. This is accomplished as follows. I_d is compared to the user-specified programming current $I_{d,prog}$ in order to obtain an error value. This error value is clamped and then input into a proportional-integral-derivative (PID) controller in order to obtain the updated V_{cg}. Figure 6.7 shows the measured characteristics of this programmer. The top subplot shows the current, which in this case was held to a target value of 1μA. The bottom subplot shows how the control gate was adjusted in order to maintain a constant current while programming.

The convergence loop in Fig. 6.6 causes the injection rate to decrease as the control gate approaches its target value, $V_{cg,target}$. The error $V_{cg} - V_{cg,target}$ is scaled by a user-defined "Rate" parameter, which controls the rate of programming. Based upon (6.4), the injection rate is exponentially dependent on V_{sd}, so we take the log of our desired injection rate in

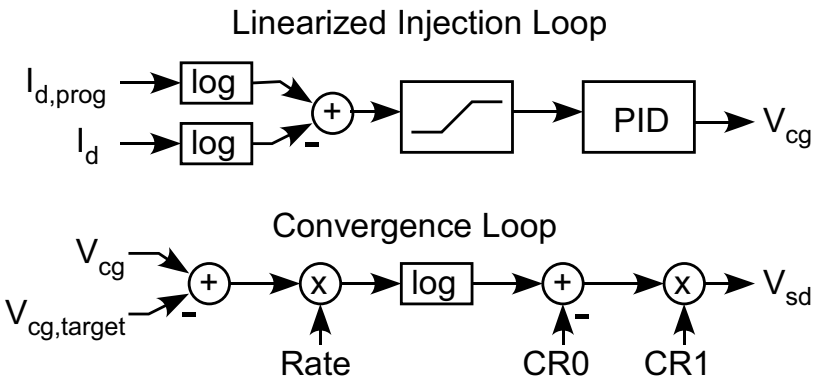

Figure 6.6: Coarse programming mode block diagram. The linearization loop is used to maintain a constant injection current and thus a constant injection rate. The convergence loop is used to slow down the linearized injection rate as the target is approached.

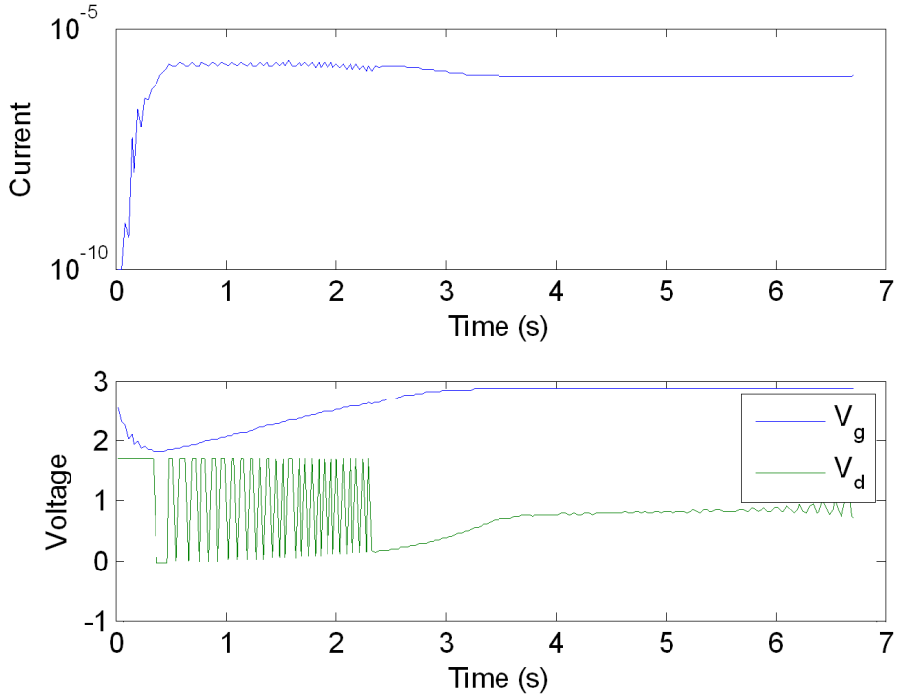

Figure 6.7: Programming a single FG in coarse programming mode. V_g is adjusted to maintain a constant current of 1μA. The charge that is injected onto the FG causes V_g to rise. As V_g approaches its target value, V_d is increased to slow down injection, as can be seen from the decreasing slope of V_g.

order to obtain the updated value of V_{sd}. Parameters CR0 and CR1 are measured curve fits that relate V_{sd} to $\log(I_{inj})$. Combining these pieces, the controller adjusts V_{sd} such that the

rate of injection decreases linearly as the distance from the target decreases. As a result, the target is approached with a linear time constant that is proportional to 1/Rate. This operation is shown in Fig. 6.7.

6.4.2 Fine Programming Mode

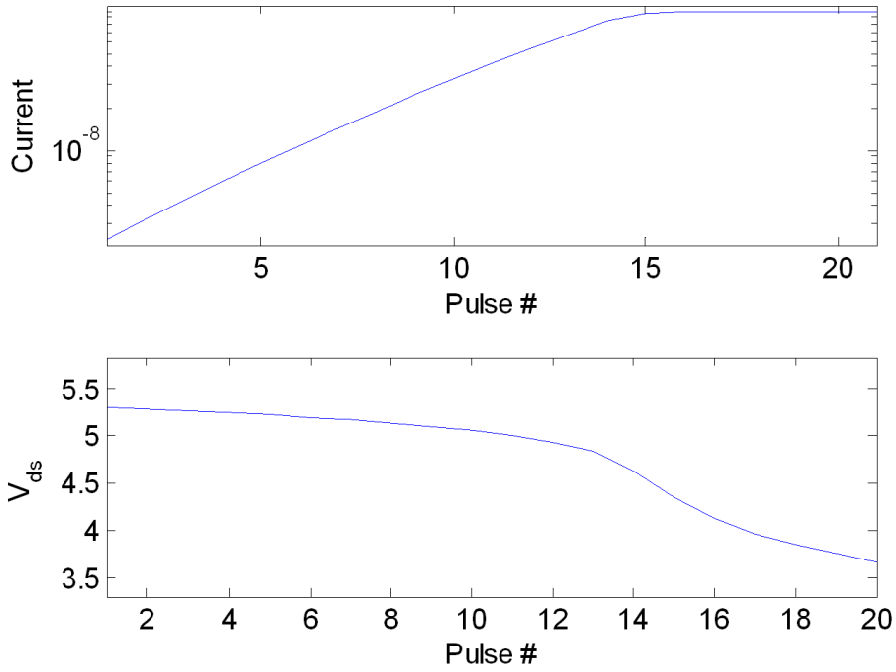

Figure 6.8: Programming a single FG in fine programming mode. Notice how the source-to-drain voltage is reduced as the current approaches its target, thus causing the current to converge more slowly.

The coarse programmer does not achieve a high enough accuracy because the FG is measured in a state that is dissimilar to its run-mode state—with high program voltages applied to the FG, and with different current levels. To obtain higher accuracy, the programmer switches to a more traditional pulse-based mode once the FG is near its target. In this fine programming mode, the FG's current is measured under the same conditions as will be applied during run mode, i.e. the same V_{dd} and the same V_{cg}. The fine-programming control algorithm works as follows. The difference between the measured current and the target current is used to determine the V_{sd} value for the next programming pulse, similar to the convergence loop in Fig. 6.6. Furthermore, in order to linearize injection, V_{cg} is updated to maintain a constant drain current for each programming pulse; this is done by calculating V_{cg} using the capacitive coupling relationship in (6.1) and also using the gate voltage/drain current relation for a MOSFET. The results of the fine programming mode are shown in Fig.

6.8. Here we can see how the source-to-drain voltage is reduced as the current approaches its target value.

Our benchtop pulsed-programming systems attains excellent accuracy, but is unusable for in-the-field programming in resource-constrained applications such as sensor networks. Consequently, we have focused more on building the low-overhead continuous-time programming system that is described in the remainder of this Chapter.

6.5 Continuous-Time Floating Gate Programming

Continuous-time FG programming is accomplished by using feedback to stop programming when the memory cell reaches its target value. A variety of continuous programming circuits have been presented: ranging from a single-transistor circuit [136] which self-converges due to the negative feedback of injection current from the FG to the drain, to more complex circuits with improved speed and accuracy. In order to linearize the characteristics of injection/tunneling, most programming circuits use feedback to maintain a constant FG voltage, though the details vary from circuit to circuit: [139] presents a three-transistor memory cell plus a comparator to stop injection once the target is reached; [140] presents a programming circuit which uses a differential FG amplifier to achieve linear tunneling and also to cancel out the tunneling junction's capacitive coupling; [141] builds upon the basic cell in [139] to create a fully integrated continuous-time FG programming system; [142] presents a memory cell which uses both hot-electron and hot-hole injection in order to converge bidirectionally toward the target. In these prior linearized techniques, the programming rate is held constant, and once the target is reached, programming is stopped; such programming faces a severe tradeoff between programming speed and accuracy [141]. In contrast, we adjust the source current to reduce the programming rate as the target is approached in order to achieve a better tradeoff between programming speed and accuracy.

Figure 6.9 shows four fundamental continuous programming configurations. The two circuits shown in Fig. 6.9(a) and (b) do not have feedback to linearize programming, but they do have inherent feedback that causes them to self-converge. In both circuits, the injection of electrons onto the FG causes V_{fg} to decrease, which decreases V_{sd} and thereby decreases I_{inj} according to (6.4). The circuit in Fig. 6.9(a) [136] can be programmed to different targets either by using different values of I_1 for a constant V_{cg}, or by using different values of V_{cg} for a constant I_1. While this circuit is very compact and produces repeatable results, its convergence time depends on the initial condition: if the initial V_{fg} is too high to supply I_1, then the small initial drain current yields little injection. As a result, convergence can take several seconds. The circuit in Fig. 6.9(b) has the opposite problem: whereas the circuit in (a) begins slow and finishes fast, the circuit in (b) begins fast and then takes minutes to slowly converge.

The long convergence times of the simple configurations can be addressed by using negative feedback to V_{cg}—shown in Fig. 6.9(c) and (d)—so that all terminals of M_{fg} are constant, and thus the rate of injection/tunneling is held constant; however, these memory cells no longer converge on their own, but require additional programming circuitry. In both of these circuits, V_{fg} is constant and V_{cg} ramps linearly up during injection, or down during tunneling, to compensate for the change in charge on the FG—see Fig. 6.9(e) and (f). V_{cg} thus

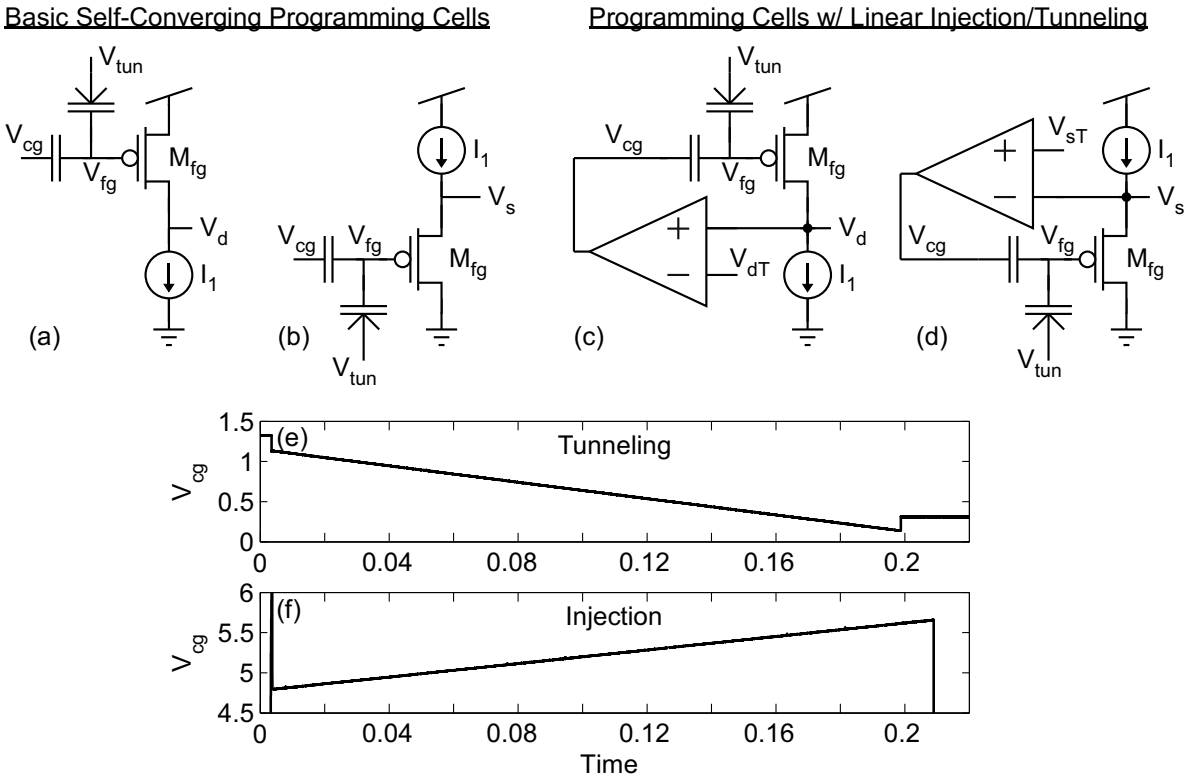

Figure 6.9: Canonical forms of programming cells. (a) and (b) are single-branch circuits without explicit feedback to keep all of the terminals of M_{fg} constant, thereby permitting situations that result in slow injection. (c) and (d) employ negative feedback to the gate to hold the terminals and current of M_{fg} constant, thus resulting in linear injection and tunneling. (e) and (f) demonstrate the linear programming characteristics of the circuit in (d).

provides our measure of the charge on the FG. We have found the high gain around the loop of the circuit in (c) to cause stability problems, and so we will not consider it any further. The source follower circuit in (d) is the same configuration that has been used in pulse-based source-feedback injection to achieve 13-bit precision with program times on the order of 50sec/200mV [138]. This circuit has good stability and offers good control over injection and tunneling through the manipulation of both V_{sT} (which sets V_s) and I_1. Our memory cell has the same basic characteristics as this circuit, but is smaller, which is important for large array applications.

6.6 Current-Conveyor-Based Memory Cell

We present a compact FG cell for continuous programming, which, when combined with our simple programmer circuit, converges to target voltages with 8-bit accuracy within 100ms.

Our basic memory cell uses the FG transistor in a source-follower configuration and

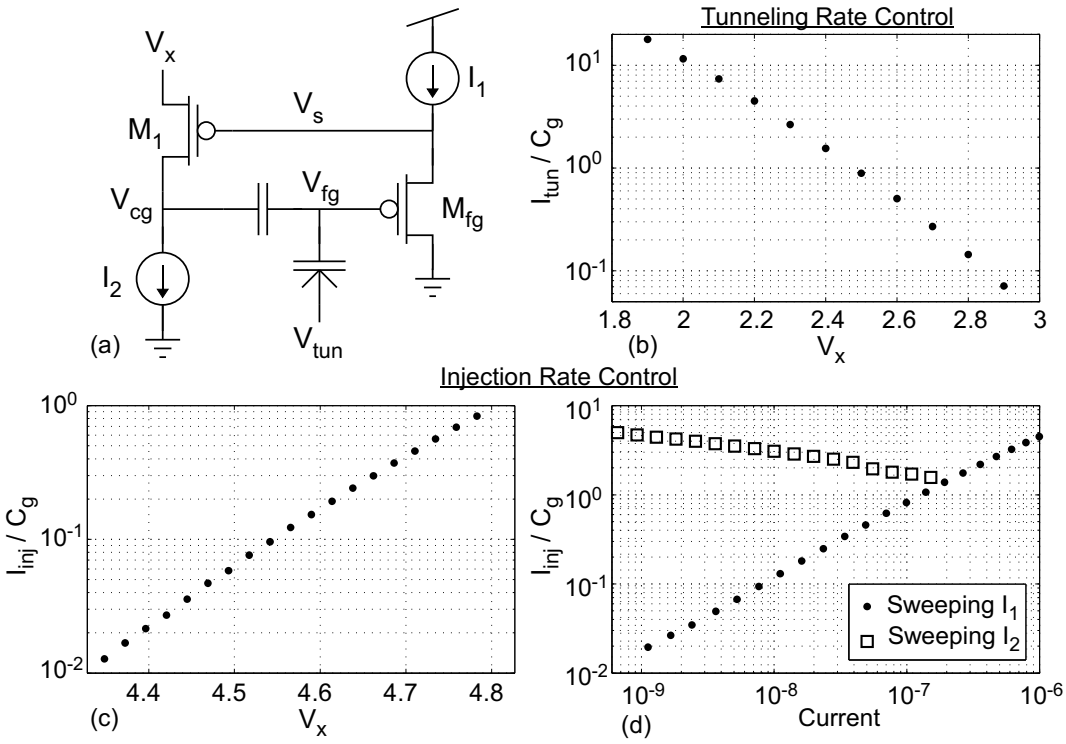

Figure 6.10: (a) Our floating-gate memory cell, which is based on the current-controlled current conveyor circuit. (b) Measured dependence of tunneling current on terminal V_X. (c)–(d) Measured dependence of injection current on the three control terminals of the circuit: V_X, I_1, and I_2.

linearizes injection via negative feedback to the control gate, as in Fig. 6.4. Such linear source-feedback injection has been used previously in [138], but we accomplish the same characteristics with the smaller current conveyor circuit that we introduce in Section 6.6. In addition to being smaller, this current conveyor memory cell also offers more flexible control over the injection rate since V_s can be modified using either a voltage or a current input.

In order to achieve the good characteristics of the circuit in Fig. 6.9(d), but reduce the size, we have developed the circuit in Fig. 6.10(a). For simplicity, current sources are shown for I_1 and I_2, but in the actual implementation, each source is implemented by a single transistor. In this memory cell, the inverting amplifier M_1–I_2 replaces the op-amp in Fig. 6.9(d). The resulting circuit structure is the current-controlled current conveyor, the details of which can be found in [143]. In this circuit, the negative feedback adjusts V_{cg} in order to force both V_{fg} and V_s to fixed voltages. The equilibrium point for V_s is controlled by both the voltage V_X and the current I_2. The equilibrium point of V_{fg} depends on both V_s and I_1. Thus we maintain independent control of the source current and drain-to-source potential (the two main injection parameters) with this four-transistor circuit. The memory cell also has linear tunneling characteristics, as illustrated in Fig. 6.10(b). In this Figure, V_X and I_1 were held fixed, while I_2 was swept from 800nA to 16μA. As a result of the increasing current, V_s, and therefore, V_{fg}, are reduced. Lowering V_{fg} causes an increased tunneling oxide voltage, and therefore an increased tunneling rate, which can be seen from the slope

of V_{cg}.

This memory cell offers three control terminals for modifying injection: two currents (I_1 and I_2) and one voltage (V_X). Using the subthreshold injection approximation in (6.4), we can solve for the injection current as a function of the control terminals in subthreshold operation

$$I_{inj} \approx \beta {I_1}^{\alpha} \left(\frac{I_2}{I_0}\right)^{-\frac{U_T}{\kappa V_{inj}}} e^{\frac{V_x-(1-\kappa)V_{dd}}{\kappa V_{inj}}} \tag{6.5}$$

where I_0 is the pre-exponential current scaler for M_1, κ is the subthreshold slope for M_1, and U_T is the thermal voltage. Figure 6.10(c)–(d) shows measured injection rates as a function of each of these control terminals. The floating-gate transistor was fabricated in a 0.35μm standard CMOS process, and has dimensions $\frac{W}{L}$=$\frac{1.6\mu m}{0.6\mu m}$ and C_g=$60fF$. All other transistors were ALD1105 FETs. The injection rate was measured by determining the slope of V_{cg} during injection experiments that were similar to Fig. 6.9(f); this slope is equal to the injection current normalized by the control gate capacitance C_g. When not being swept, V_X, I_1, and I_2 were held fixed at 5V, 860nA, and 2nA, respectively. Additionally, since the feedback holds V_{fg} constant, this cell has linear tunneling characteristics. Figure 6.10(b) shows the dependence of the tunneling current on V_X while all other terminals were held fixed. The experiments shown in Fig. 6.10(b)–(d) demonstrate the ability to adjust the cell's programming rate over a large range using either voltage or current inputs. Additionally, the weak dependence on I_2—approximately an inverse fifth root dependence—makes I_2 appropriate for fine rate adjustment. Furthermore, the cell works well in the subthreshold region, where power consumption is low and (6.5) holds true.

6.7 Programmer Circuit

The combination of control terminals makes the memory cell very flexible in terms of programming circuits. Figure 6.11(a) illustrates one possible programming circuit, which uses I_1 as the control terminal. The transconductor converts the difference between V_{cg} and the target value, V_{targ}, into a current. This current is rectified by the current mirror M_2–M_3 and is forced into the source terminal of the FG transistor. As the target is approached, the injection rate is reduced, and eventually stopped, by the reducing I_1. Figure 6.11(b) shows a timing diagram of the programming process. During the pre-injection interval (i), the supply voltage is at the run-level value (3V), and the FG has been tunneled to the point that V_{cg} is at ground. When the injection interval (ii) first starts, the supply voltage is ramped up (5.4V), which pulls V_{cg} up, and there is a short time during which the capacitance of node V_{cg} is discharged through I_2; the significant duration of this discharge time is due to the fact that the circuit was prototyped on a breadboard with significant parasitic capacitance. Once V_{cg} is discharged, we observe linear injection while the transconductor's output current is saturated. As V_{cg} approaches the target, I_1 is reduced (see the bottom pane of the plot), and once V_{cg} reaches the target, I_1 is zero. During interval (iii), the current conveyor structure has stopped operating due to I_1 being shut off, and as a result, V_{cg} is pulled high. Then for the read mode interval (iv), the supply is ramped down to run level, and the cell is configured as a voltage reference. The cell's voltage output is read from V_{cg}, and the cell is

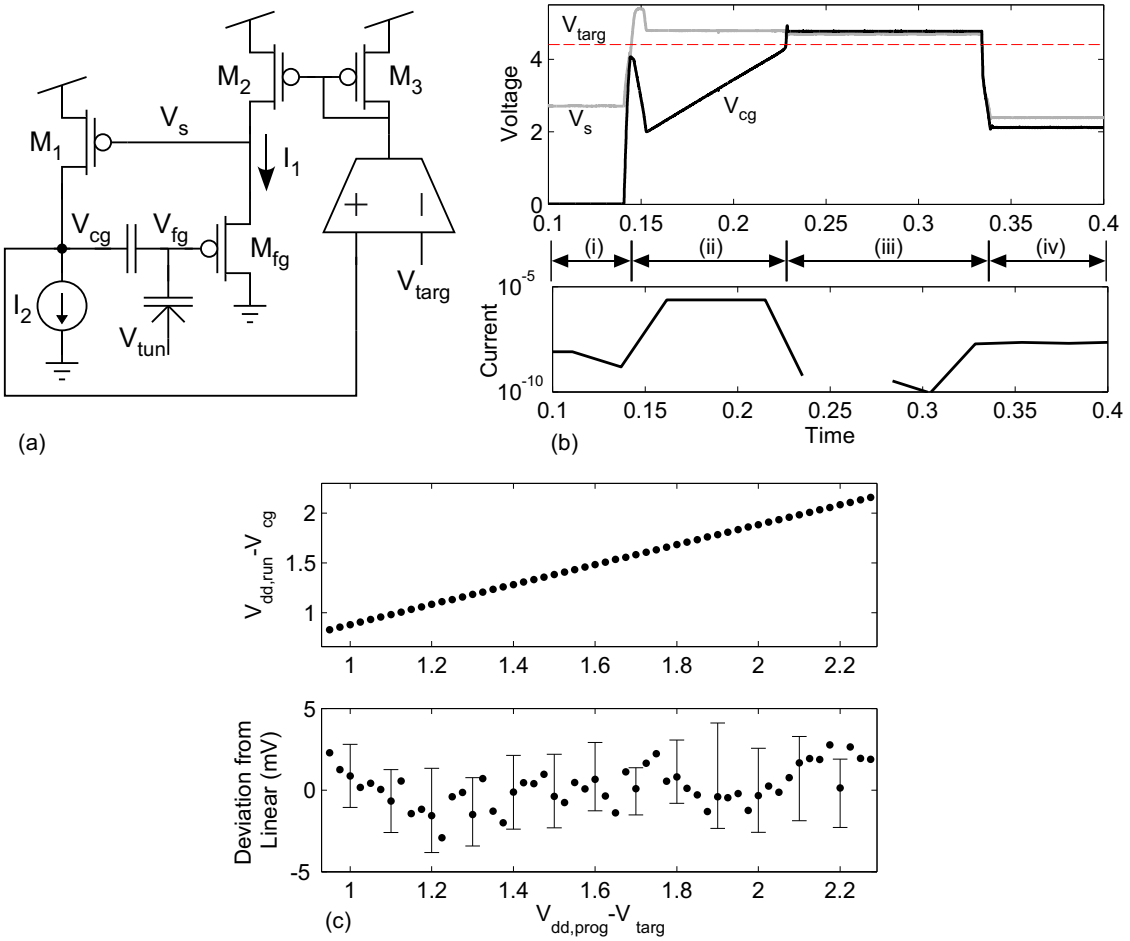

Figure 6.11: (a) Our memory cell programming circuit. (b) Measured timing diagram of the programming circuit, showing the control gate, source, and target voltages as well as the source current. While V_{targ} is greater than V_{cg}, the floating-gate injects and V_{cg} rises. As the target is approached, I_1 is reduced, slowing injection until the target is reached, at which point the current is shut off. Afterwards, the supply voltage is reduced to a run-mode level, a constant current is applied for I_1 in order to bias the current conveyor, and the programmed value can then be read from V_{cg}. (c) Measured programming accuracy.

configured by removing the transconductor from the loop and supplying a constant voltage to the gate of M_2 (I_2 remained constant throughout the experiment). Alternatively, the FG can be disconnected from the memory cell and placed into a separate circuit for a current output.

Figure 6.11(c) shows the results of performance experiments on the memory cell and programmer combination. A standard wide output-range transconductor was used. The memory cell was programmed to linearly spaced values of V_{targ}, and the value of V_{cg} was measured after the circuit was placed in read mode—for which the supply voltage was ramped down, the transconductor was disconnected, and a fixed current was applied for I_1. The top pane shows that the memory cell has a linear relationship between V_{targ} and the ramped

down V_{cg} (with a slope of 1.003 and an offset of 122mV). The deviation from a straight line is shown in the bottom pane. For every fourth data point, the memory cell was programmed 100 times in order to verify repeatability; from these data are derived the error bars which show maximum and minimum values. Over a range of 1.36V, the maximum deviation is 4.2mV, thus yielding an accuracy of over 8-bits. Note that this prototype was built on a breadboard with discrete components, and for the low current levels that were used, a noise floor on the order of millivolts was observed.

The current values used for run mode were I_1=20nA and I_2=2nA, yielding a power consumption of 66nW/cell when configured as a voltage reference. These currents do not need to be exact and do not require precise matching across cells—but the currents should be stable. During programming, the transconductor bias is set to 2μA, and the maximum program power consumption is 43μW/cell.

6.8 Array Architecture

Since an advantage of FGs is that they allow for dense analog memory arrays, an FG memory cell should be suitable for placement within an array. In Fig. 6.12, we show a two-by-two array of our memory cell. In order to save space, the program control circuit is shared amongst the cells in a column; this facilitates simultaneous programming of all cells in a row.

The procedure for programming row 0 is as follows. The programming circuits are connected to the array by setting PROG to high; voltage output is selected by setting VO to high; and row 0 is selected by setting "Row 0 Select" to high, thus connecting row 0's voltage output and control input (gate of M_{I1}) to the column programming circuit. For the unselected rows, the gate of M_{I1} is pulled to V_{dd} to prevent injection. In run mode, the memory cell can be used either with voltage outputs or with optional current outputs. The current outputs are accessed by pulling VO low so that the drain of M_{FG} goes out through the current output and also by pulling the source of M_{FG} up to V_{dd} by lowering the Column I_1 lines. With V_s high, M_X is off and the control gates of the cells are connected to V_A by raising the I_2 biases.

6.9 Conclusion

We have presented a compact analog FG memory cell. We have also presented a continuous-time programmer circuit for the memory cell which achieves 8-bit accuracy with 100ms program times. An integrated implementation of this programming scheme with 300 memory cells is used in our FPAA that is described in Chapter 10. Details on that integrated programmer are provided in Section 10.3.

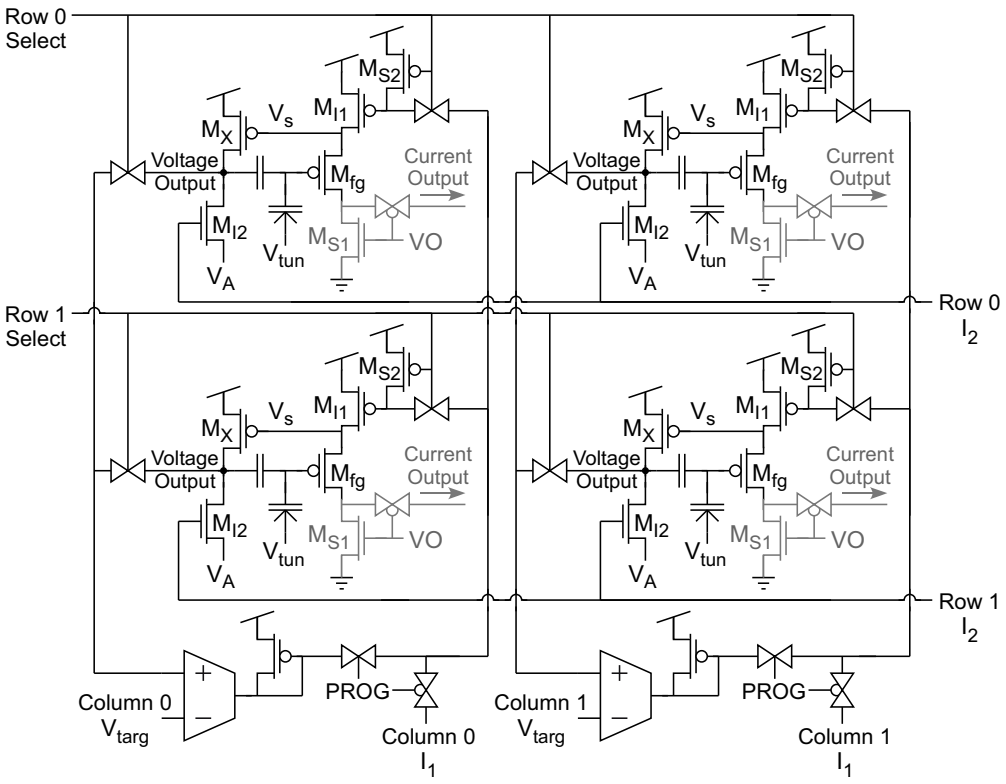

Figure 6.12: Array architecture for the memory cell and programmer.

Chapter 7

Modeling of Charge Manipulation in Floating-Gate Transistors

Due to the widespread use of floating-gate (FG) transistors in Flash memory, much research on improving the density and reliability of FGs has been incorporated into highly customized Flash-memory fabrication processes. But as Systems-on-Chips scale to increased complexity, and as analog systems include greater numbers of tunable parameters, the need for on-chip nonvolatile memory has made it necessary to port FG transistors from custom memory processes into generic CMOS processes. Accordingly, we have performed detailed characterizations of the charge manipulation processes—Fowler-Nordheim tunneling and hot-electron injection—in CMOS FG transistors. These characterizations have helped us to optimize the design of individual FG transistors, and have also helped us to improve the accuracy of our FG simulation model [144], which is crucial for designing floating-gate transistor circuits. In this Chapter, we present our methodology and results, and apply these results to FG transistor design.

Section 7.1 examines the mask design of the floating-gate's tunneling junction, where erasure and/or writing occur. Aided by static and transient tunneling current measurements for a variety of tunneling junctions, we present recommendations for constructing tunneling junctions to minimize the duration, power consumption, and oxide degradation of programming. This work was published in Electronics Letters [145].

Section 7.2 presents characterization of hot-electron injection. These characterizations are used to extract parameters for a compact equation that models injection in FG transistors with different dimensions and in different processes.

7.1 Efficiency and Reliability of Fowler-Nordheim Tunneling in CMOS Floating-Gate Transistors

In FG transistors, erasure—and often writing—is achieved by tunneling electrons through the tunneling junction, C_{tun}. The design of this junction has significant implications on the speed, efficiency, and long-term reliability of writing and erasure. In this Section, the characteristics of the two basic tunneling junction structures in standard CMOS are compared, and the junction size that achieves minimal tunneling duration and oxide degradation is derived.

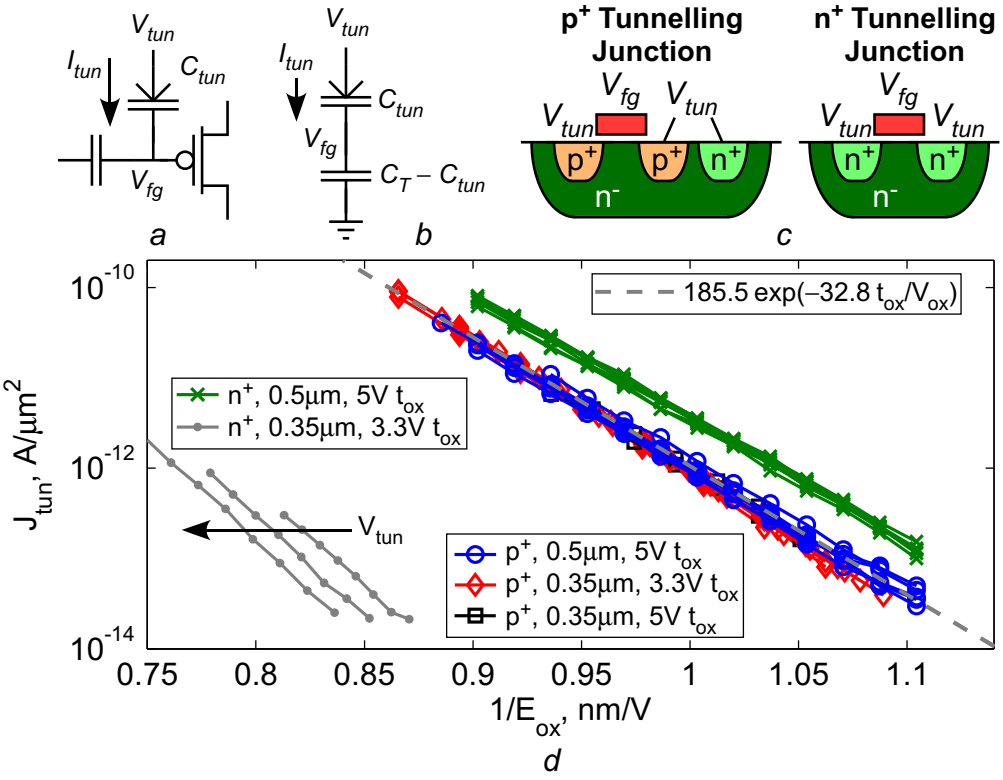

Figure 7.1: Fowler-Nordheim tunneling characteristics. (a) Schematic of floating-gate transistor. (b) Equivalent tunneling circuit. (c) Structure of tunneling junctions. (d) Fowler-Nordheim voltage-current measurements.

7.1.1 Fowler-Nordheim Tunneling Current

Fowler-Nordheim tunneling occurs when the electric field across the C_{tun} dielectric is sufficiently high to distort the energy band such that the effective barrier thickness is reduced to 5nm [146]. An electric field (E_{ox}) of 0.64V/nm is required to initiate tunneling for the 3.2eV Si-SiO_2 interface (from the floating gate to the oxide dielectric) [146]. The complete expression for Fowler-Nordheim tunneling into SiO_2 is derived in [147]. By neglecting temperature dependence, by neglecting oxide barrier-lowering from image charge (which is small at moderate fields in SiO_2 [148]), and by dropping the pre-exponential electric field term (which, within the region of interest, only has an affect on the curve fit values), the tunneling current expression can be approximated by [149]

$$J_{tun} = \alpha \exp\left(-\beta t_{ox}/V_{ox}\right) \tag{7.1}$$

where t_{ox} is the thickness of the oxide barrier, V_{ox} is the voltage across the barrier, and α and β are constants related to the fabrication process and junction type. A thin oxide is desired to minimize the tunneling voltage. In standard CMOS, the gate oxide is typically used because it is thin and also of high quality, which benefits reliability and predictability. Oxides thinner than 5nm should be avoided to deter direct tunneling; consequently, higher voltage I/O devices for 2.5V (5nm), 3.3V (7–8nm), or 5V (14–15nm) operation are typically

used in fine-geometry processes [137]. Thus, our results using 3.3V and 5V devices from 0.35μm and 0.5μm processes provide a relevant insight into tunneling in new processes, as well.

To remove electrons from the FG, V_{tun} is raised to a high voltage, typically higher than the reverse breakdown voltage of the source/drain diffusions, but less than the breakdown of the well-to-substrate junction. To avoid reverse breakdown, tunneling junctions are generally placed within a well. Fig. 7.1(c) shows the two basic types of tunneling junctions: a p^+ MOS capacitor formed as a standard pFET and an n^+ MOS capacitor formed with n^+ diffusions along the gate. The n^+ junctions have traditionally been favoured for analog memory applications [149], but p^+ junctions are becoming common for standard CMOS Flash applications [137]. In this Section, the static and transient characteristics of p^+ and n^+ junctions are compared to determine recommendations for junction design.

FG programming characteristics can be engineered via the design of the tunneling junction: the width, length, t_{ox}, and diffusion type. Fig. 7.1(d) shows measured Fowler-Nordheim tunneling characteristics for a variety of junction designs. Each trace was obtained by reading V_{fg} through a buffer during a pulse to V_{tun} (i.e. typical tunneling conditions). All terminals except V_{tun} and V_{fg} were held fixed, so the circuit can be modeled by Fig. 7.1(b). By reading V_{fg}, we obtain $E_{ox} = (V_{tun} - V_{fg})/t_{ox}$ and $I_{tun} = C_T \frac{d}{dt}(V_{fg})$, where C_T is the total capacitance connected to the node. Four different C_{tun} dimensions were used on a 0.5μm process (μm$\times\mu$m): 1.5×0.6, 3×0.6, 1.5×1.2, and 3×1.2. Five dimensions were used for the 0.35μm p^+ junctions (μm$\times\mu$m): 0.5×0.5, 0.5×1, 0.5×2, 1×0.5, and 2×0.5. The 0.35μm n^+ junction was 0.4μm×0.35μm.

The p^+ junction curves in Fig. 7.1(d) all align and are excellently described by the values α=185.5A/μm^2 and β=32.8V/nm. The traces align when normalized by area (i.e. plotted as current density), which illustrates that, at least for large enough V_{ox} to achieve fast tunneling, the current comes from the full junction area rather than from the edges [149].

The curves for the n^+ junctions correspond to the time after which the junctions have recovered from depletion and have begun to tunnel (more details in the next Subsection). For 0.5μm, the difference between the p^+ and n^+ junctions is likely caused by their different flat-band voltages. The low current and V_{tun}-dependence of the 0.35μm n^+ junction may be explained by variations in the effective oxide thickness due to finite charge depth [150].

In summary, p^+ junctions are more consistent from process to process and the p^+ tunneling current is significantly higher in the 0.35μm process.

7.1.2 Temporal Dynamics of Tunneling Junctions

In addition to the Fowler-Nordheim tunneling traces, the temporal dynamics of the junctions must also be considered. The variable capacitance of the MOS capacitor structures can cause complex transient characteristics. Fig. 7.2*a* shows measured transient responses of 0.5μm FGs for 20V V_{tun} pulses. This experiment is analogous to block erasure in which all FGs, regardless of their initial value, should tunnel to approximately the same value. The pulse duration for the p^+ junctions is 340μs and the durations for the n^+ junctions have been adjusted to achieve an approximately equal amount of tunneling. Based on the equivalent circuit in Fig. 7.1(b), V_{fg} will rise as electrons tunnel through C_{tun}. As V_{fg} rises, the tunneling rate decreases due to a decreasing V_{ox}. As a result, FGs with different ini-

tial voltages approach the same final voltage [see the p^+ junctions in Fig. 7.2(a)]. The p^+ junctions perform as expected given (7.1). The n^+ junctions, however, experience a voltage-dependent delay before they begin to tunnel. This delay is a result of the depletion region that is formed underneath the gate in response to the V_{tun} pulse. Most of the tunneling voltage is dropped across the depletion capacitance, resulting in a small oxide voltage and thus no tunneling current. The depletion region collapses slowly as carriers are generated from thermal generation and band-to-band tunneling, after which tunneling begins [151]. In both processes, the p^+ junctions had no measurable delay.

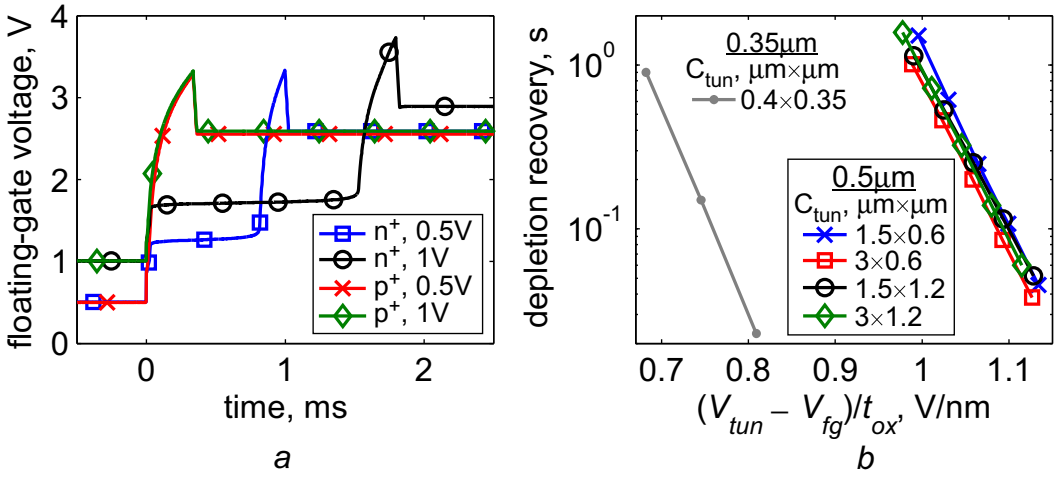

Figure 7.2: Tunneling junction transient characteristics. (a) Transient characteristics of n^+ and p^+ junctions with different initial voltages. (b) Depletion recovery time of n^+ junctions.

Fig. 7.2(b) shows that the depletion recovery time of the n^+ junctions (duration between the beginning of the tunneling pulse and the start of tunneling) is independent of the junction area when plotted in terms of $V_{tun} - V_{fg}$. However, larger junctions have a smaller $V_{tun} - V_{fg}$ due to the capacitive division [Fig. 7.1(b)]; as a result, the delay time increases with C_{tun}. Another result of the voltage-dependence is that the delay time is exponentially related to the initial FG voltage. This is problematic for memory arrays because, for short erase times, the post-erase distribution of FG values can have a complex and non-monotonic relation to the initial distribution of FG voltages. These transient depletion characteristics are more pronounced for typical tunneling voltages in the 0.5μm process than in the 0.35μm process. But in both processes, the p^+ junctions achieve faster tunneling times because of their higher I_{tun} in 0.35μm and because of their lack of a depletion recovery delay.

Overall, we suggest that p^+ junctions are the best choice because they are faster, less power to operate (since they are faster, the high-voltage generation circuitry operates for less time), more consistent from process to process, and always available in CMOS process design kits.

To verify reliability using high-voltage tunneling pulses, we have performed 100k write/erase cycles on a 0.35μm FG with a p^+ tunneling junction and a 200fF gate capacitor. The FG's threshold voltage was shifted 1V up and down in each cycle, transferring an accumulated 20nC of charge through C_{tun}. We observed only a 30% reduction in tunneling current.

7.1.3 Sizing of Tunneling Junctions for Speed and Reliability

Tunneling junctions are often made to be minimum size to minimize the coupling from V_{tun} to V_{fg}. However, we will derive the optimal C_{tun}/C_T ratio that minimizes the time to tunnel to the post-erasure FG voltage, $V_{fg,e}$. Increasing the junction size has two opposing trends: larger area increases the tunneling current, but it also increases the coupling onto the FG which reduces the final voltage when V_{tun} steps down. To find the junction size that tunnels to the final voltage in the shortest duration, we first write the tunneling current in terms of C_{tun} and $V_{fg,e}$ as

$$I_{tun} = \alpha(C_{tun}/\gamma)\exp\left[-\frac{\beta t_{ox}}{\left(1-\frac{C_{tun}}{C_T}\right)V_{tun}-V_{fg,e}}\right] \tag{7.2}$$

where γ is the unit capacitance ($aF/\mu m^2$) of C_{tun}. By taking the derivative with respect to C_{tun} and setting the LHS to '0', we find that tunneling is maximized for the following coupling ratio.

$$\frac{C_{tun}}{C_T} = \frac{\beta t_{ox}}{2V_{tun}}\left(1+2\frac{V_{tun}-V_{fg,e}}{\beta t_{ox}} - \sqrt{1+4\frac{V_{tun}-V_{fg,e}}{\beta t_{ox}}}\right) \tag{7.3}$$

This equation is verified in Fig. 7.3 for 0.5μm FGs. For V_{tun}=20V, $V_{fg,e}$=2.5V, t_{ox} ≈14nm, and β=32.8V/nm, the optimal coupling ratio is calculated to be approximately 3.1%. It can be seen that the junction with 3.2% coupling reaches the final voltage 23% faster than the larger junction (which suffers from excessive coupling) and 44% faster than the smaller junction (which suffers from insufficient tunneling area, thus limiting I_{tun}). In addition to reducing the duration compared to a minimum-sized tunneling junction, the larger sizing also increases the long-term reliability. This is because oxide degradation is related to the charge-density that has passed through the junction [152]. By using a larger junction, the charge-density is reduced, which contributes to an increase in long-term reliability. For digital Flash applications, C_T may not be large enough to achieve such a ratio, but analog FG applications use large control-gate capacitors, often 100fF or more, and so will benefit from this sizing.

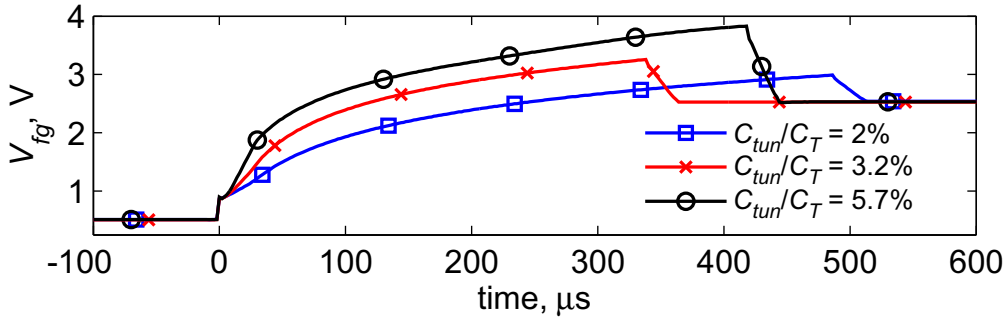

Figure 7.3: Optimal tunneling junction sizing.

7.1.4 Conclusion of Tunneling-Junction Study

Non-volatile memory is increasingly being included in standard CMOS products. Tunneling is used in most of these non-volatile memories, and so design methods for tunneling junctions are of interest if they can improve speed, reliability, and/or energy efficiency. We have presented answers to tunneling junction design decisions that offer improvement in all three categories.

7.2 Characterization of Hot-Electron Injection Across Varying Transistor Dimensions

CMOS FGs, particularly analog FGs, are often programmed using hot-electron injection. In non-FG transistors, injection can still occur and is undesirable because of the damage that constant injection can cause to the gate oxide. As a result, extended drain implants are added to nFETs to minimize injection by reducing the electric field at the drain. Such prevention measures are not added to pFETs, because the lower mobility and shorter mean-free path of holes already inhibits injection in pFETs [153]. Nevertheless, pFETs are preferred for injection in CMOS FGs because they do not have extended drains.

Modeling of injection is more complex than modeling of tunneling because injection is a multivariate phenomena (i.e. it depends upon both the channel current and the channel-to-drain electric field) and because injection is defined by a larger number of process parameters. Most injection modeling has focused on nFETs, which further complicates injection modeling for CMOS FGs wherein pFETs are preferred over nFETs. To improve modeling of CMOS FGs, we present injection data from 30 FGs with varying widths, lengths, oxide thicknesses, and junction depths. We show that two extracted parameters are sufficient to describe these data.

7.2.1 Injection Measurement

Figure 7.4(a) shows our setup for measuring injection. The FG transistor is placed in a linearized loop, as described in Chapter 6. The loop maintains constant I_d and constant V_{sd}; consequently, I_{inj} is constant. To obtain I_{inj}, the voltage at V_{cg} is measured during the experiment. Electrons injected onto the floating gate give the FG voltage a more negative charge, which the linearization loop counteracts by raising V_{cg}. As a result, $I_{inj} = C_g \frac{dV_{cg}}{dt}$.

Figure 7.4(b) shows the measurement of one value of I_{inj}. The slope of V_{cg} is used to obtain I_{inj} for one pair of I_d and V_{sd} values. A full injection characterization of a FG transistor is obtained by repeating this measurement for multiple values of I_d and V_{sd}, which are controled by V_1 and V_2, respectively. An ammeter is used to measure I_d at each value of V_1. The op-amp forces $V_{sd} = V_2$. The full injection characterization for one FG transistor is shown in Fig. 7.4(c). In the remainder of this Section, we describe a method for parameterizing injection in pFETs.

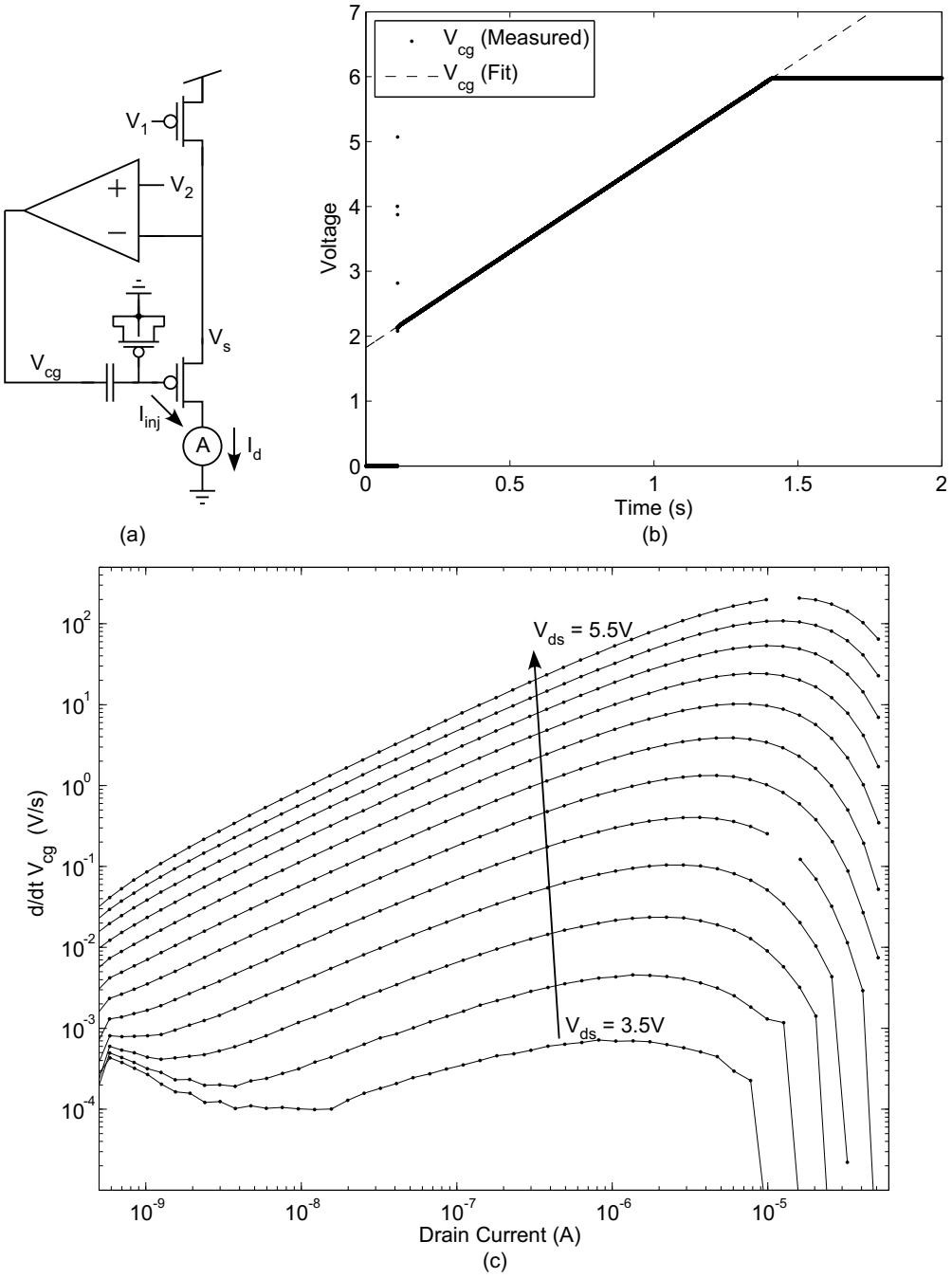

Figure 7.4: Methodology for characterizing injection. (a) Measurement setup. The injection current is obtained by measuring the change in V_{cg} under constant drain current and source-to-drain voltage. (b) Measurement of the change in V_{cg}. The slope is extracted to obtain the injection current. (c) Measured injection rate at various drain currents and source-to-drain voltages.

7.2.2 Injection Parameterization

Hot-electron injection in pFETs is a multi-event process [149]. First, a hole crossing the channel is accelerated in the high-field region of the drain. Second, this hot hole impacts

the lattice and thus releases a hot electron. And third, this hot electron is swept over the gate-oxide potential-barrier to the more positive voltage of the gate. Each event relies on the previous and has a limited probability of occurring.

Various models of the injection process have been developed [154, 155]. We have obtained the best matching to experimental data using the "lucky-electron model," which was developed by Shockley in 1961 for p-n junctions [156], applied to hot-carrier gate current in nFETs by Hu in 1979 [157, 158], and then extended to pFETs by Ong in 1990 [154].

In the lucky-electron interpretation, each carrier has a limited probability of causing the chain of events that lead to an electron being injected onto the gate. Therefore, increasing the channel current increases the injection current by simply increasing the number of injection opportunities. Additionally, increasing the source-to-drain voltage will increase the electric field near the drain (while the device remains in saturation) and will thus increase the probability that a hole will create a hot electron. The last key to facilitate injection is that the gate-to-drain voltage should be high enough to sweep the hot electrons over the gate-oxide potential barrier.

Before we introduce the lucky-electron equation, let us revisit the subthreshold-region injection approximation that we used to simplify hand calculations in Chapter 6:

$$I_{inj} \approx \beta {I_s}^{\alpha} e^{V_{sd}/V_{inj}} \tag{7.4}$$

where I_s is the source current, and β, α, and V_{inj} are device-dependent fits [136,159]. In Fig. 7.4(c), the subthreshold region is approximately the range where $I_d < 1\mu$A. Since the slope, α, is a weak function of V_{sd} in the subthreshold region, (7.4) is valid if V_{sd} is constrained to a limited range. This equation is consistent with the interpretation of injection that was described above: greater current tranlates to more opportunities for injection and greater source-to-drain voltage translates to higher probability of injection. However, (7.4) is clearly not sufficient for all operating regions.

The equation that we use for lucky-electron injection, shown below in (7.8), comes from [160]; here we summarize the elements that make up the equation. The form of the lucky-electron equation is [154]

$$I_{inj} = \gamma I_d E_m \exp\left(-\frac{\delta}{E_m}\right) \tag{7.5}$$

where γ and δ are fits that are invariant to the fabrication process and to the dimensions of the device, and E_m is an approximation for the maximum field at the drain. The maximum field can be calculated as [160]

$$E_m = \frac{V_{sd} - V_{sd,sat}}{l} = \frac{V_{gd} + V_{T0}}{l} \tag{7.6}$$

where $V_{sd,sat}$ is the saturation voltage, V_{T0} is the threshold voltage, and l is the length of the velocity saturation region near the drain, which can be calculated as [160]

$$l = 0.22 t_{ox}^{1/3} x_j^{1/2} \tag{7.7}$$

where t_{ox} is the gate oxide thickness and x_j is the diffusion-area junction depth, both of which can be obtained from simulation model files. Combining these yields the expression

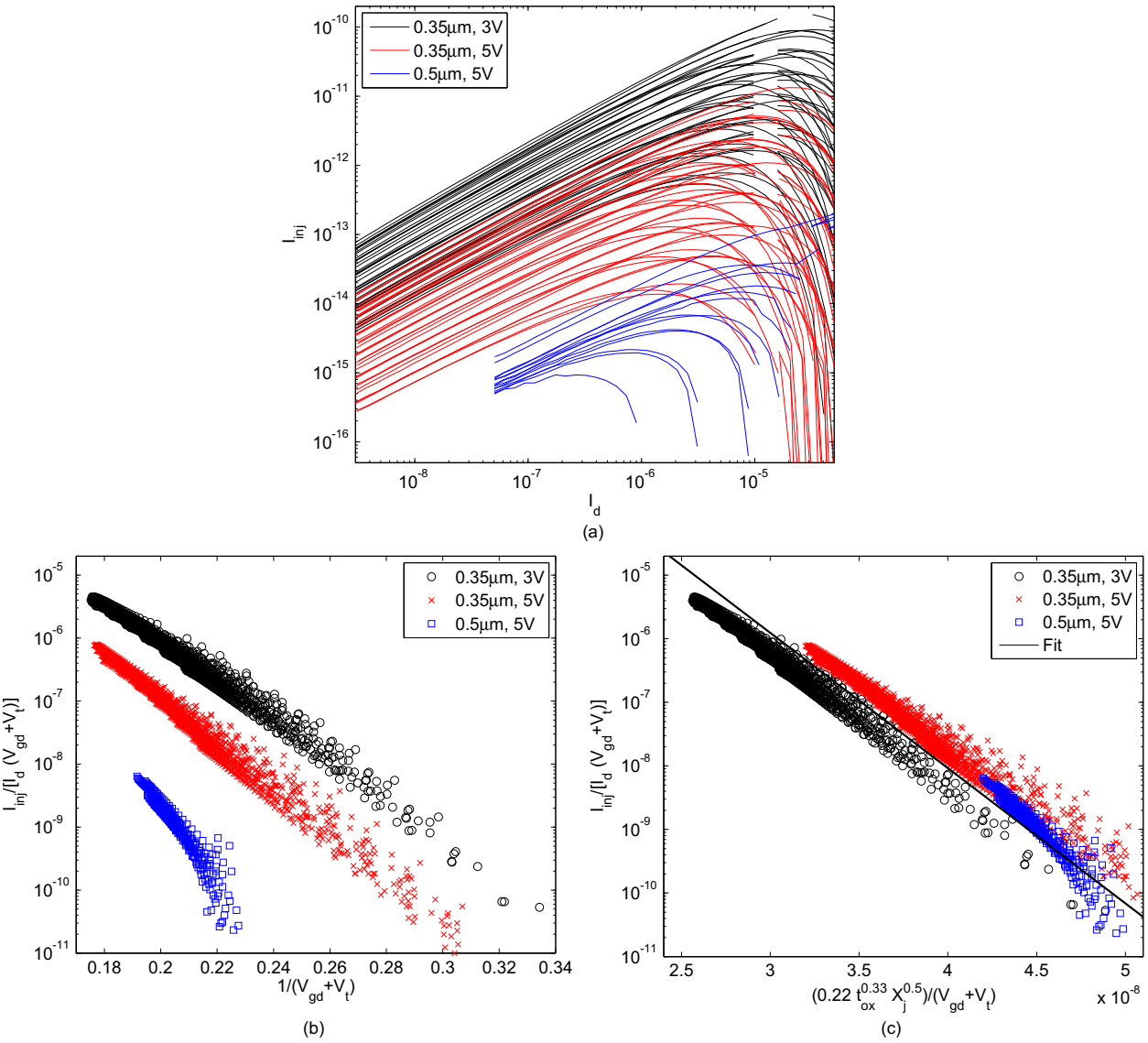

Figure 7.5: Extraction of injection parameters. (a) Raw injection current data for 30 FG transistors with various dimensions and in different processes. (b) Injection curves normalized for transistor W and L and also for V_{sd}. (c) Injection curves further normalized for oxide thickness and junction depth. The fit uses (7.8) with $\gamma = 3$ and $\delta = 4.9 \times 10^8$.

that we use to model injection

$$I_{inj} = \gamma I_d \frac{V_{gd} + V_{T0}}{0.22 t_{ox}^{1/3} x_j^{1/2}} \exp\left(-\frac{\delta 0.22 t_{ox}^{1/3} x_j^{1/2}}{V_{gd} + V_{T0}}\right) \tag{7.8}$$

To extract the parameters γ and δ, we characterized 30 FG transistors:

1. In a 0.35μm CMOS process: Seven different transistor dimensions (W/L) were used for the FGs (μm/μm): 2/0.5, 2/1, 2/2, 1/0.7, 4/0.7, 8/0.7, and 8/2. For each size,

both a thin-oxide (3V) and a thick-oxide (5V) device are included for a total of 14 devices.

2. In a 0.5μm CMOS process: Sixteen different transistor dimensions (W/L) were used for the FGs (μm/μm): 3/0.6, 6/0.6, 12/0.6, 24/0.6, 3/1.2, 6/1.2, 12/1.2, 24/1.2, 3/2.4, 6/2.4, 12/2.4, 24/2.4, 3/4.8, 6/4.8, 12/4.8, and 24/4.8. The 0.5μm FGs were all thin-oxide (5V) devices.

The 0.5μm devices were measured at V_{sd}=5V. The 0.35μm devices were measured for six linearly spaced V_{sd} values from 4.6V to 5.5V.

All of the measured data are shown in their raw form in Fig. 7.5(a). These data are first normalized by transforming them into the form of (7.8), but without normalizing for l. This normalization is achieved by plotting the data using $1/\left(V_{gd}+V_{T0}\right)$ for the x-axis and using $I_{inj}/\left[I_d\left(V_{gd}+V_{T0}\right)\right]$ for the y-axis. The result is shown in Fig. 7.5(b). Each of the three device types cluster into straight lines, regardless of the source-to-drain voltage and regardless of the width (W) and length (L) of the transistor. If V_{sd} and I_d are held fixed, then increasing W/L will increase the injection current because a smaller value of V_{sg} will be required, which will thus increase V_{gd}. However, when the data are visualized in terms of V_{gd}, then the dependence on W/L disappears.

The final step in extracting the injection parameters is to normalize across device types. In Fig. 7.5(b), the two 0.35μm devices have the same junction depth x_j, and the two 5V devices have approximately the same oxide thickness t_{ox}. By further normalizing the data by l, all three devices cluster on a single line. Consequently, devices with varying size and across different processes can be described by a single equation, (7.8), and by two parameters, $\gamma = 3$ and $\delta = 4.9 \times 10^8$. The only other parameters (t_{ox}, x_j, and V_{T0}) can be obtained from the device model. As a result, it is simple to incorporate injection into floating-gate simulation models, such as [144].

7.3 Conclusion and Future Work

In this Chapter we have presented detailed characterization of the charge-manipulation mechanisms in FG transistors. These mechanisms are not modeled in circuit simulators, so our results can help circuit designers to more accurately simulate FGs. Additionally, these results are beneficial for designing the dimensions of FG transistors.

An important characteristic that will be studied in future work is the accuracy of injection-based programming. High levels of programming accuracy can be obtained by applying ever smaller injection pulses as the programming target is approached. However, future work should examine the accuracy limitations at various injection rates by determining the noise component of the injection current at arbitrary operating points.

Chapter 8

A Regulated Charge Pump for Programming Floating-Gate Transistors

In Chapter 6, we presented a nonvolatile analog memory cell that is based upon floating-gate transistors, and which can be used to create low-power programmable analog systems for embedded applications. This memory cell, as well as any other memory cell that is based upon floating-gate transistors, requires write and erase voltages that exceed the nominal supply voltage. To use analog floating-gate transistors in resource-constrained applications, such as wireless sensor networks, these program voltages should be generated within the analog signal processor, and should be impervious to the unstable supply voltages that are often found in such applications. To accommodate these needs, we present a regulated charge-pump step-up converter in this Chapter. This charge pump is used to generate tunneling (i.e. erasure) voltages in our field-programmable analog array in Chapter 10.

8.1 Floating-Gate Programming Voltages in Standard CMOS

The most common form of solid-state nonvolatile memory is the floating-gate transistor—a CMOS-compatible device that is the basis of Flash memory [133], and that can also be used to create dense, low-power, programmable analog parameters [161]. Floating-gate transistors store information in the form of a trapped charge on an electrically-floating polysilicon gate. Under nominal operating voltages, this charge will remain on the gate. To program a different amount of charge onto the gate, higher voltages are used to induce channel hot-electron injection and Fowler-Nordheim tunneling, both of which enable electrons to pass to and from the gate via the gate oxide. These processes require voltages that exceed the chip's nominal operating voltage, but which should be generated on-chip "as needed" to minimize power and complexity in embedded systems. Figure 8.1 illustrates the generation of programming voltages for a floating-gate memory array. The programming voltages are greater than the rated voltage of the core devices—even greater than the source/drain breakdown voltages. Consequently, the voltage generators must be designed to ensure reliability, in addition to

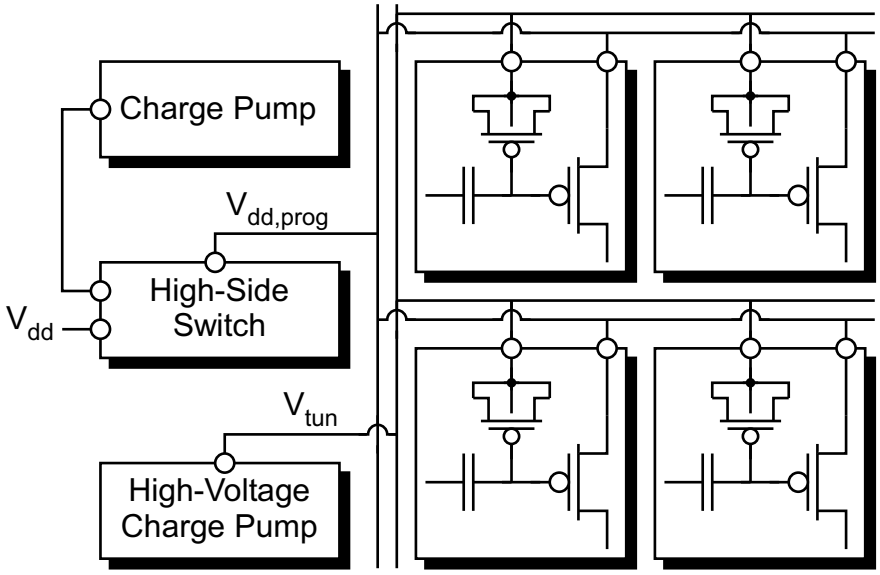

Figure 8.1: The use of charge-pump step-up converters to generate write ($V_{dd,prog}$) and erase (V_{tun}) voltages for a nonvolatile memory array.

minimizing the energy consumption and size.

As described in Chapter 7, the efficacy of the charge-manipulation processes (i.e. hot-electron injection and Fowler-Nordheim tunneling) are functions of the gate-oxide thickness (t_{ox}) and the source-/drain-junction depth (X_j). Figure 8.2(a) shows how these features scale in standard CMOS [162]. Charge retention in floating-gate transistors is compromised by direct tunneling when $t_{ox} < 5$nm [146]. This fact is born out by the continued use of 6–7nm gate oxides in bleeding-edge NAND Flash processes while logic processes have continued scaling to 0.8nm gate oxides [163]. Consequently, floating gates that are built in standard logic processes below the 250nm node should use thick-oxide I/O devices ($t_{ox,FG}$) that are rated for operation at 2.5V or greater.

Figure 8.2(b) shows the scaling of floating-gate programming voltages in standard CMOS. The tunneling voltage (V_{tun}) was calculated for 1ms erase times using (7.1). V_{tun} stops scaling at the 250nm node because of the transition to thick-oxide devices to maintain low-leakage operation. The injection supply voltage ($V_{dd,prog}$) was calculated for 50ms write times using (7.8). Despite the stagnant oxide thickness, $V_{dd,prog}$ continues to scale because of X_j. Note that V_{tun} is always greater than the drain-to-body breakdown voltage (V_{jbkdn}), which indicates the difficulty of generating V_{tun} using the available devices in a given CMOS process.

To generate the programming voltages, step-up converters must be used. To minimize energy consumption, these converters should only be enabled when they are needed. The step-up converters will typically be powered by the chip V_{dd}. The step-up ratios for tunneling and injection are shown in Fig. 8.2(c). The step-up ratio is lowest for the 250nm through 600nm nodes because 1) below the 250nm node, higher-voltage I/O devices are used to make low-leakage floating gates and 2) above the 600nm node, devices are operated at 5V even though they can accommodate much higher voltages. However, the rated V_{dd} of a technology

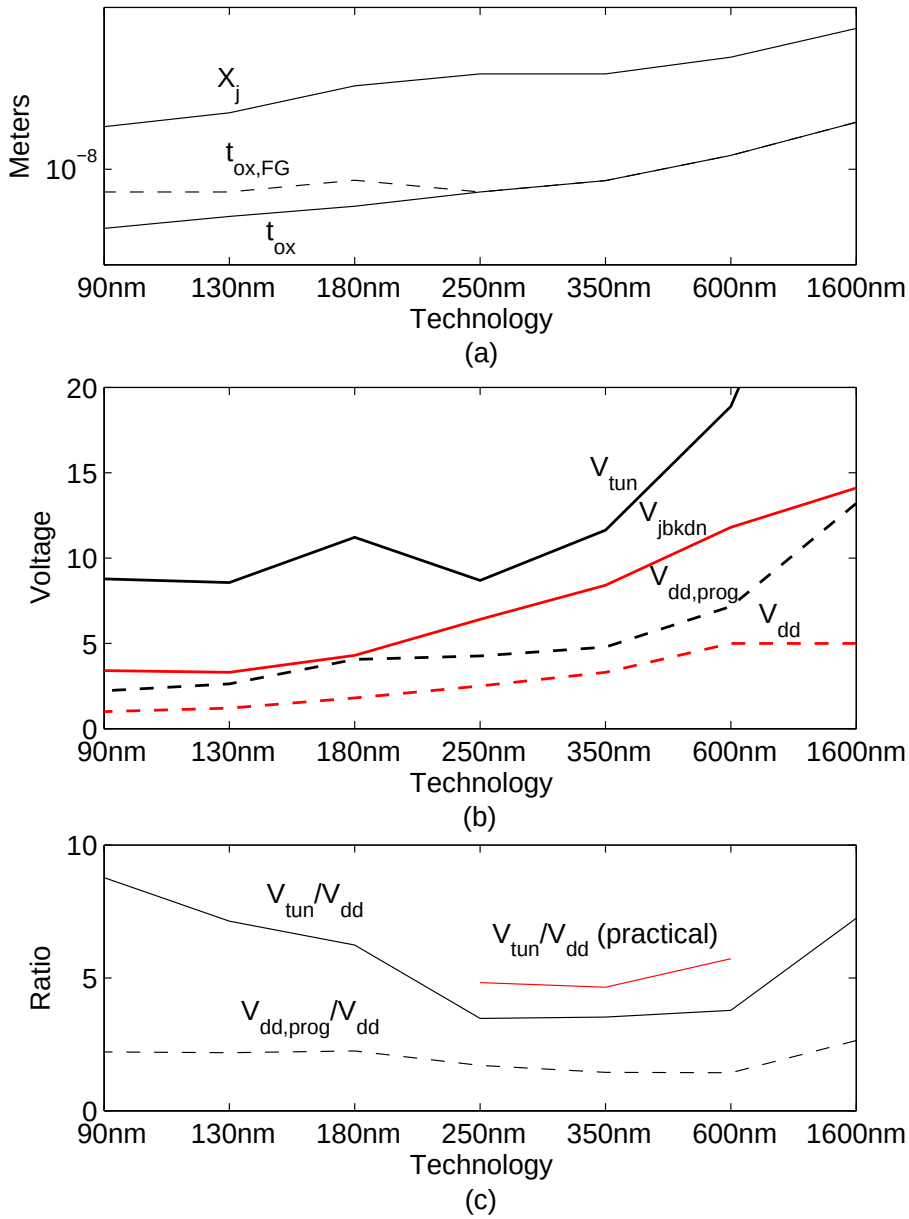

Figure 8.2: Scaling of write and erase voltages in standard CMOS. (a) Scaling of junction depth (X_j) and oxide thickness (t_{ox}) [162]. To minimize leakage in floating-gate transistors, thicker oxide I/O devices are used to keep $t_{ox,FG} > 5$nm. The programming voltages are determined by these parameters. (b) Scaling of critical programming voltages: the core supply voltage (V_{dd}), the write (i.e. injection) voltage ($V_{dd,prog}$), the drain-to-body breakdown voltage (V_{jbkdn}), and the erase (i.e. tunnel) voltage (V_{tun}). (c) Ratio of the write ($V_{dd,prog}$) and erase (V_{tun}) voltages to the core supply voltage.

node causes too much hot-electron injection for stable floating-gate operation. Consequently, the 250nm through 600nm nodes must use lower supply voltages in practice, so the ratios for these nodes are essentially one integer value higher than shown in the Figure.

In summary, Fig. 8.2 provides some constraints on the design of voltage step-up converters for floating-gate programming. Additional constraints that are specific to embedded analog applications include small size with minimal external components, fast start-up and shut-down, and consistent voltage generation in the presence of potentially noisy supply voltages. Voltage consistency is especially important since it affects the programming accuracy. In the remainder of this Chapter, we describe our design of an integrated high-voltage charge pump for generating the tunneling voltage. The charge pump is regulated to generate a consistent voltage for accurate programming. This charge pump has been fabricated in a 0.35μm standard CMOS process without high-voltage add-ons. This charge pump can easily be adapted to generate the injection voltage; we have focused on generating the tunneling voltage because the high step-up ratio and the operation beyond the junction-breakdown voltage makes it more difficult to generate tunneling voltages.

8.2 Overview of Charge Pump Circuitry

8.2.1 Charge Pump Topologies

A "charge pump," sometimes called a "voltage multiplier," is a type of switched-capacitor circuit that is used for voltage conversion. To step up a voltage V_a, a charge pump first samples V_a onto a capacitor and then raises the bottom plate of the capacitor by V_b, at which point the voltage of the top plate of the capacitor is $V_a + V_b$. By alternately sampling voltages onto multiple capacitors, the voltage can be increased in a succession of stages.

Some charge pump topologies have exponential voltage growth as the number of stages increases [164], but the charge-transfer switches in these topologies are subjected to exponentially higher voltages in each stage. This voltage stress is a limitation, and as a result, charge pump topologies with linear voltage growth are more common [165]. The Cockcroft-Walton charge pump (which Cockcroft and Walton built for their Nobel-prize winning experiments in which they disintegrated the atomic nucleus [166]) has linear voltage growth and is commonly used for charge pumps that are built from discrete parts. However, the Cockcroft-Walton topology connects the capacitors in series, making it highly sensitive to stray capacitance on the bottom plates, which can be large in integrated circuits—the Cockcroft-Walton topology is thus inappropriate for integration [167].

Figure 8.3(a) shows an idealized Dickson charge pump [167], which is the standard charge pump topology for integrated circuits because of its linear voltage growth and its insensitivity to stray bottom-plate capacitance. The charge pump shown in the Figure has two stages that are clocked by alternating clock phases ϕ_1 and ϕ_2. The sequential operation of the charge pump is illustrated in Fig. 8.3(b). In the first stage, when ϕ_1 is open and ϕ_2 is closed, node V_1 is charged to V_{dd}. Since ϕ_1 is low at this time, the voltage across the first pumping capacitor (C_{p1}) is V_{dd}. Then ϕ_2 opens and ϕ_1 goes high, thus raising the bottom terminal of C_{p1} to V_{dd}. Since C_{p1} has sampled V_{dd}, V_1 is raised to $2V_{dd}$. This process repeats in the second stage, where C_{p2} is charged by C_{p1} while ϕ_1 is closed. Although C_{p1} and C_{p2} form a capacitive divider that attenuates the voltage that was sampled onto C_{p1} in the previous cycle, in steady-state, C_{p2} has charge from the previous cycle such that V_2 is sampled at

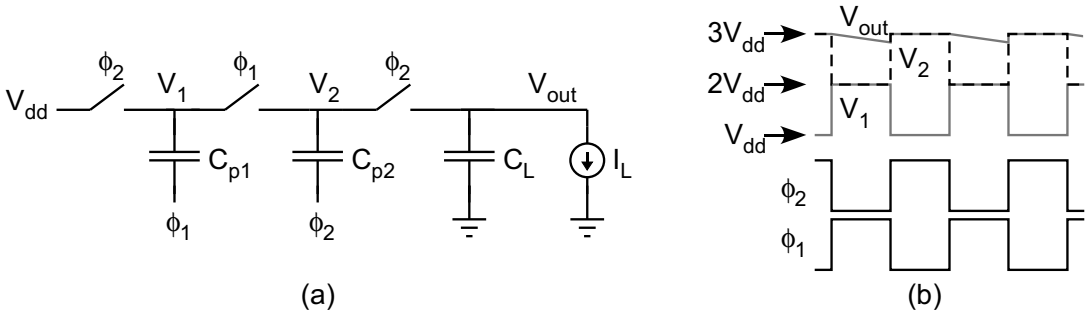

Figure 8.3: (a) Ideal charge pump. (b) Operation of the ideal charge pump.

$2V_{dd}$.[1] Then ϕ_2 goes high and V_2 is raised to $3V_{dd}$. The final output voltage is obtained by sampling the last stage onto the load capacitor C_L. Higher voltages can be generated by cascading more stages (N). Each stage adds V_{dd} to obtain a total open-circuit output voltage of $V_{out} = (N+1)V_{dd}$.

When a load current (I_L) is drawn from the charge pump, V_{out} is reduced. In equilibrium, the load draws a charge of $I_L T$ during each cycle of duration T. During a cycle, each pumping capacitor replenishes this charge to its successor. As a result, the voltage to which each capacitor pumps is reduced by $I_L T/C_p$. This lost voltage accumulates for each capacitor and yields a total voltage of

$$V_{out} = (N+1)V_{dd} - N\frac{I_L}{C_p f} \tag{8.2}$$

where $f = 1/T$ is the pumping frequency [168]. A nonideal charge pump will have additional sources of voltage loss caused by the "on" resistance of the switches and also caused by the stray capacitance of the switches, which forms a capacitive divider with the pumping capacitors. However, these losses can be minimized by designing the switches to have a small voltage drop for the expected load current and by designing the pumping capacitors to be much larger than the stray capacitance.

8.2.2 Charge Pump Regulation

When a charge pump is used to generate write/erase voltages, V_{out} must be stable and consistent to facilitate accurate floating-gate programming. However, it is evident from (8.2) that V_{out} has an amplified dependence on the supply voltage V_{dd}, which may be inconsistent and noisy in a battery-powered sensor node with a duty-cycled radio. V_{out} also has a

[1]For example, on the j-th cycle of ϕ_1, the voltage at V_2 is the superposition of the voltage that is sampled at V_1 on the previous half-cycle (which is raised by V_{dd} when the bottom plate goes high) and the voltage at V_2 on the previous cycle

$$V_2(j) = \frac{C_{p1}\left(V_1(j-1/2) + V_{dd}\right) + C_{p2}V_2(j-1)}{C_{p1} + C_{p2}} \tag{8.1}$$

The circuit is designed with $C_{p1} = C_{p2}$ and V_1 equals V_{dd} on the half cycles. If the circuit is in equilibrium, then $V_2(j) = V_2(j-1) = 2V_{dd}$.

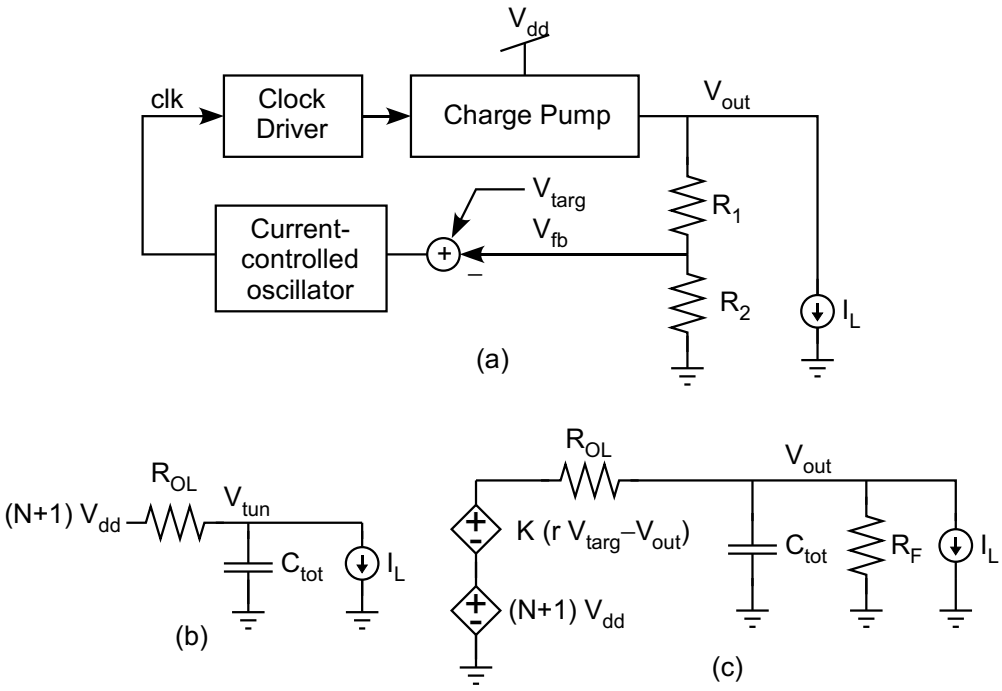

Figure 8.4: (a) Block diagram of a regulated charge pump. (b) Linear model of an unregulated charge pump. (c) Small-signal model of a regulated charge pump.

dependence on the load current I_L, which may vary as memory cells begin and finish programming. Furthermore, V_{out} is set to integer multiples of V_{dd}, which is a limitation for setting the sensitive program voltages to optimal values. To achieve reliable programming, the charge pump should be regulated to a constant output voltage.

Examining (8.2), only two quantities can be adjusted at runtime to regulate the output voltage: V_{dd} and f. Regulation using V_{dd} does not actually modulate the supply voltage, but instead modulates the clocking voltages ϕ_1 and ϕ_2 [169, 170]. These clocking voltages contribute the NV_{dd} portion of V_{out}. These variable-pump-voltage regulators have the advantage of reducing the level of clock-feedthrough ripple on V_{out}, which must otherwise be removed with a large load capacitor. However, variable-pump-voltage regulators have the disadvantage that they constantly operate at their maximum frequency, which results in wasted power from unnecessarily charging and discharging all parasitic capacitances. Variable-frequency regulators are thus a more efficient alternative. The simplest type of variable-frequency regulator is the "skip" regulator, which "turns on" a constant-frequency oscillator when V_{out} is less than the desired voltage and otherwise turns the oscillator off [171]. Some regulators have used a combination of variable-voltage and skip-mode [172,173]. Skip regulators exhibit sporadic bursts of pumping which create large ripple on the output. A better alternative is a true "variable-frequency" regulator that linearly increases or decreases the frequency to regulate V_{out}. Aaltonen and Halonen previously used this method of regulation [174], and we have used the same basic method. Our charge pump achieves higher efficiency, smaller size, and better load regulation.

A generic block diagram for a variable-frequency regulated charge pump is shown in

Fig. 8.4(a). A voltage-divider ($R_{1,2}$) reduces the output voltage to the chip's rated voltage range. The difference between this reduced voltage and the desired voltage (V_{targ}) is used to modulate the pumping frequency until the output voltage locks onto the desired value. In addition to setting the output voltage, regulation also reduces the output resistance, increases the power-supply rejection, and shortens the start-up time compared to an open-loop charge pump.

The performance of an open-loop charge pump can be obtained by modeling it as the RC circuit shown in Fig. 8.4(b). From (8.2), the open-circuit voltage is $(N+1)V_{dd}$ and the open-loop output resistance is $R_{OL} = N/(C_p f)$. The total capacitance at the output (C_{tot}) combines the true load capacitance (C_L) with the distributed charge pump capacitance $C_{eq} = NC_p/3$ [168]. The open-loop performance parameters are summarized in Table 8.1.

Table 8.1: Charge Pump Performance

	Open loop	Closed loop
Output resistance	$R_{OL} = N/(C_p f)$	$R_{CL} = R_{OL}/K$
Power supply rejection	$\text{PSRR}_{OL} = 1/(N+1)$	$\text{PSRR}_{CL} = K\text{PSRR}_{OL}$
Start-up time constant	$\tau_{OL} = R_{OL}C_{tot}$	$\tau_{CL} = \tau_{OL}/K$

It is more difficult to determine the closed-loop regulation performance because of R_{OL}'s dependence on the operating point (i.e. R_{OL} depends upon f, which is a function of V_{out} and I_L), which makes the regulation loop nonlinear. To simplify the analysis, the small-signal model from [174] is adapted and shown in Fig. 8.4(c). The variables are all defined in Table 8.2. The dc voltage source has been replaced by two small-signal dependent sources. The bottom source models the circuit's sensitivity to V_{dd}. The top source models the effect of the frequency-modulating feedback. The loop gain, K, is the product of 1) the attenuation due to the voltage divider ($1/r$), 2) the voltage-to-frequency conversion gain K_F of the error amplifier and oscillator, and 3) the frequency-to-voltage conversion gain of the charge pump

$$K_{CP} = \frac{dV_{out}}{df} = \frac{I_L N}{f^2 C_p} \tag{8.3}$$

Also, the resistor R_F models the resistance of the voltage divider, which may consist of linear resistors or may be implemented with a chain of diode-connected pFETs to save space.

To solve for the regulation performance, first equate the currents at V_{out}

$$\frac{1}{R_{OL}}\left[(N+1)V_{dd} + K\left(rV_{targ} - V_{out}\right) - V_{out}\right] = V_{out}\left(sC_{tot} + \frac{1}{R_F}\right) + I_L \tag{8.4}$$

Then solve for V_{out}, noting that by design $\frac{K}{R_{OL}} \gg \frac{1}{R_{OL}} + \frac{1}{R_F}$

$$V_{out} = \frac{rV_{targ} + \frac{N+1}{K}V_{dd} - \frac{R_{OL}}{K}I_L}{s\frac{C_{tot}R_{OL}}{K} + 1} = \frac{rV_{targ} + \frac{1}{\text{PSRR}_{CL}}V_{dd} - R_{CL}I_L}{s\tau_{CL} + 1} \tag{8.5}$$

The output consists of a superposition of three components: the scaled-up target voltage, which is the desired output, as well as unwanted contributions from the supply voltage and

Table 8.2: Charge Pump Variables

N	number of stages
V_{targ}	target voltage
I_L	load current
R_{OL}	output resistance of open-loop charge pump
R_F	resistance of voltage divider
$C_{tot} = C_L + C_{eq}$	total output capacitance
C_L	load capacitance
$C_{eq} = NC_p/3$	distributed charge pump capacitance
$K = K_F K_{CP}/r$	loop gain
K_F	voltage-to-frequency gain of the error amplifier and oscillator
$K_{CP} = I_L N/(f^2 C_p)$	frequency-to-voltage gain of the charge pump
r	value of voltage division

the load current, which are both suppressed by the loop gain. The closed-loop performance parameters, which are summarized in Table 8.1, are all improved by a factor of the loop gain compared to the open-loop performance. It should be noted that the regulation circuitry adds little area and power compared to the charge pump. In Section 8.5, we will connect these performance parameters to actual circuit parameters.

In the remainder of this Chapter, we describe the subcircuits and the measured performance of the regulated charge pump.

8.3 The Charge Pump Stages

The primary challenge when designing a charge pump that approaches the ideal characteristics in (8.2) is the design of the charge-transfer switches (CTS). Early integrated charge pump designs used diodes or diode-connected transistors to implement the CTS [167]. Such designs rely upon the uni-directional current flow of diodes to only allow charge transfer onto a pumping capacitor when its voltage is exceeded by the voltage of the preceding stage. These designs suffer from poor voltage gain and poor efficiency because of the accumulation of diode voltage drops. As a result, several charge pump circuits have been developed over the past two decades to dynamically control the on/off state of the CTS and thus achieve superior performance.

As illustrated in Fig. 8.2(b), Fowler-Nordheim tunneling requires voltages so large that the drain-to-body junction will break down. To avoid break down, transistors in the CTS must inhabit isolated wells so that all voltage differentials within the CTS are at safe values. Since it is not always possible to isolate nFETs in CMOS processes, it is best for high-voltage charge pumps to use pFETs exclusively.

Most all-pFET CTS circuits are based upon the circuit shown in Fig. 8.5(b) [175]. The CTS is shown within the dashed line and consists of 1) M_{sw}, the switch that transfers charge between neighboring stages; 2) M_{bt}, a "boosting" switch that enforces the correct dc

operating point onto the otherwise capacitively-coupled gate of M_{sw}; and 3) C_{bt}, a capacitor that ac-couples the on/off clock (ϕ_{1b}) up to the higher local voltage of the CTS.

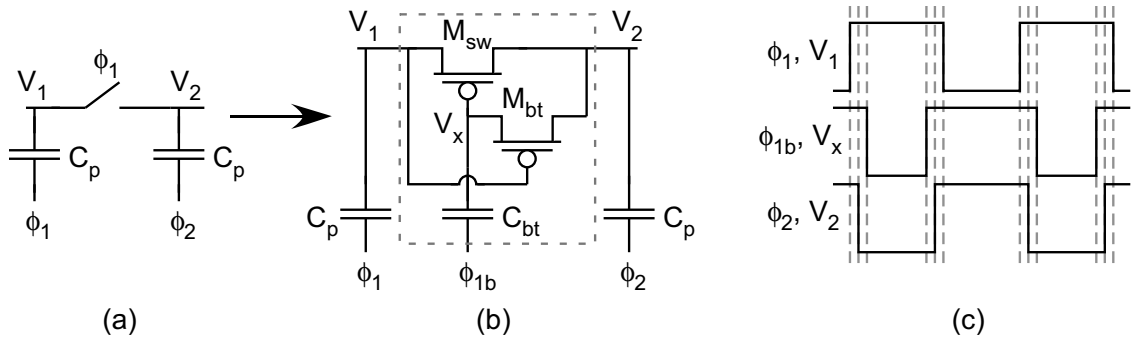

Figure 8.5: (a) Charge-transfer switch. (b) A common all-pFET charge-transfer switch. (c) Non-overlapping clock scheme for the all-pFET charge-transfer switch in (b).

Figure 8.5(c) shows the clock sequence that yields correct operation of the all-pFET CTS. First, ϕ_1 goes high to prepare for charge transfer. When ϕ_1 goes high, it shuts off M_{bt}, thus sampling the source of M_{sw} (V_2) onto the gate of M_{sw} (V_x), which keeps V_x at the correct dc operating point (i.e. the gate is referenced to the source). Next, ϕ_2 is taken low to prepare for charge transfer. ϕ_2 goes low after ϕ_1 goes high so that the correct voltage is sampled onto node V_x. To transfer charge, ϕ_{1b} is taken low to turn on M_{sw}. After charge transfer, ϕ_{1b} is raised to disconnect V_2 from V_1. ϕ_{1b} goes high before ϕ_2 so that no charge leaks backwards from V_2 to V_1.

The charge pump circuit that we use is based upon the charge pump presented by Li *et al* in [176] (simulation results only). The charge pump's CTS is similar to the all-pFET CTS in Fig. 8.5(b), but has modifications (described below) that reduce the voltage stress on the transistors and also to reduce the ripple on the output voltage. This charge pump is shown in Fig. 8.6(a).

Each stage has two parallel paths—a top path through M_{sw1} and M_{sw3}, and a bottom path through M_{sw2} and M_{sw4}. The parallel paths conduct in opposite phases, which helps to reduce the ripple. Furthermore, the opposing phases of the parallel paths offers a low-complexity means for clocking the second set of switches (M_{sw3} and M_{sw4}). This second set of switches reduces the voltage stress on the transistors in the off-phase. In a standard charge pump, the voltage across an off switch is $2V_{dd}$. By adding the extra switches in this charge pump, the off voltage is divided across the series switches so that no pair of terminals on the transistors is exposed to a voltage higher than V_{dd} [176].

To provide the correct voltage for the transistor wells, the active well-biasing technique is used [177]. This technique is implemented by "bulk-biasing" transistors M_{bb*}. Each pair of M_{bb*} transistors connects the well to the higher voltage terminal of the source or the drain. In the original implementation of this charge pump [176], active well-biasing was not used for switches M_{sw3} and M_{sw4}, but instead the wells were connected to $V_{stage,out}$. In steady-state, this was acceptable because they included a grounded capacitor at $V_{stage,out}$ to hold the higher of V_{m1} and V_{m2}. However, we have removed this capacitor and included well-biasing on these switches to avoid activating the parasitic vertical BJT during startup. Reducing

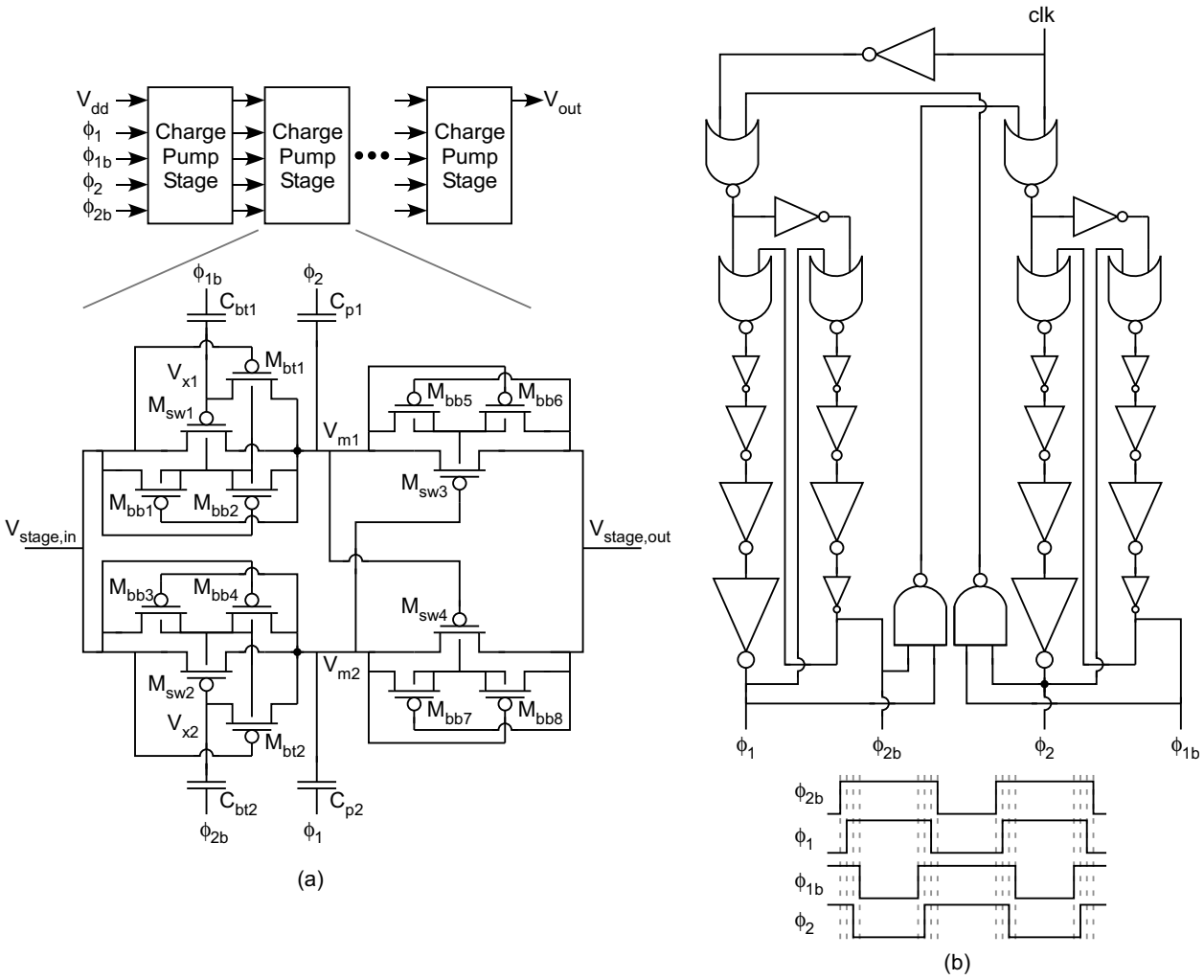

Figure 8.6: (a) The all-pFET charge pump stage [176] that is used throughout this work. (b) Our 4-phase non-overlapping clock generator circuit. Feedback of all outputs via NAND gates has been added to guarantee that ϕ_1 and ϕ_2 do not overlap.

the startup power is important for charge pumps that are used for tunneling because the startup energy dominates the overall energy for the short tunneling pulses.

We have fabricated this charge pump in a standard 0.35μm CMOS process. The size of the charge pump, including all regulation circuitry, is 230μm $\times$ 300μm (a die photograph is shown in Fig. 8.7). The design specifications are summarized in Table 8.3. The area-optimization routine in [168] was used to obtain initial values for the number of stages and C_p, which were further optimized via simulation.

Although the charge pump contains many devices, its size is dominated by the pumping capacitors C_{p*}. Each C_p in a double-branch charge pump is only half of the value that it would be in a single-branch charge pump with commensurate performance, so the total capacitance is no more than for a single-branch charge pump.

The clock generator circuit is shown in Fig. 8.6(b). In contrast to the clock generator in the previous implementation [176], which used a non-overlapping clock to trigger the two

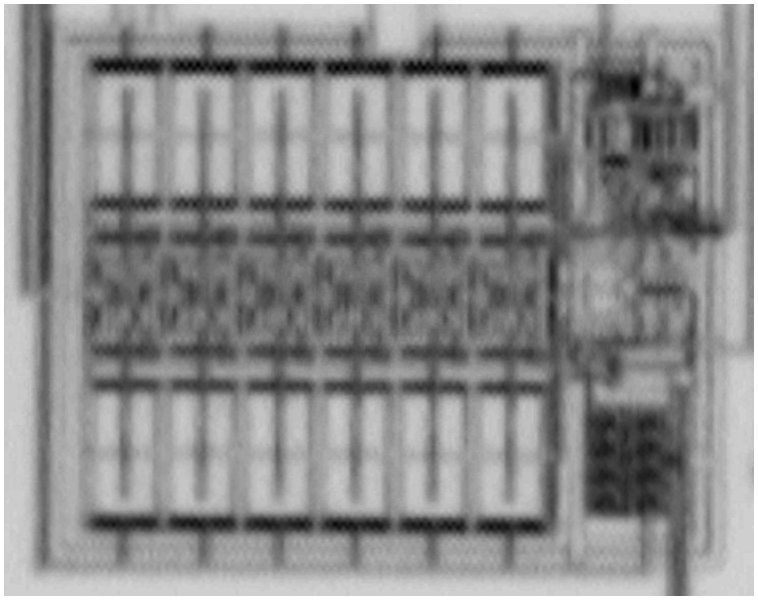

Figure 8.7: Die photograph of the complete regulated charge pump circuit. The size is 230μm $\times$ 300μm.

non-overlapping paths, we have enclosed the left and right halves of the clock generator circuit into a global loop to ensure that ϕ_1 and ϕ_2 are non-overlapping. Most of the area of the clock generator is consumed by the clock drivers for ϕ_1 and ϕ_2, which are necessary for all charge pumps.

In summary, this all-pFET charge pump fulfills the requirements of reliably and efficiently generating high voltages for Fowler-Nordheim tunneling. Measured open-loop performance is shown alongside the closed-loop performance in Section 8.5.

8.4 The Current-Controlled Oscillator and the Edgifier

In Section 8.2.2, we explained why variable-frequency regulation is a good choice for low-power, low-ripple regulation. To modulate the frequency in our variable-frequency regulator, we have designed a current-controlled oscillator. In comparison to voltage-controlled oscillators, current-controlled oscillators naturally offer linear input-to-frequency gain over a wide operating range, and are also easily limited so that the maximum frequency of the charge pump is not exceeded during transients. Our current-controlled oscillator, shown in Fig. 8.8, is based upon a three-stage current-starved ring oscillator. The frequency increases linearly with I_{in}. The current-to-frequency gain has been measured to be approximately 2kHz/nA over a range of 100Hz to 10MHz.

One of the primary attractions of variable-frequency regulation is that under light load conditions, the clock frequency reduces so that power consumption is minimized. However, the clock signals that are generated by low-frequency oscillators have long rise and fall times. These slow-moving edges create excessive short-circuit current when they are connected directly to a logic gate. Unlike the dynamic current that charges fan-out gates, this short-

Table 8.3: Charge Pump Specifications

Technology	0.35μm CMOS
V_{dd}	2.5V
# Stages	6
C_{bt}	110fF
C_p	1.5pF
M_{bb}	3μm x 0.35μm
M_{bt}	3μm x 0.35μm
M_{sw}	5μm x 0.35μm

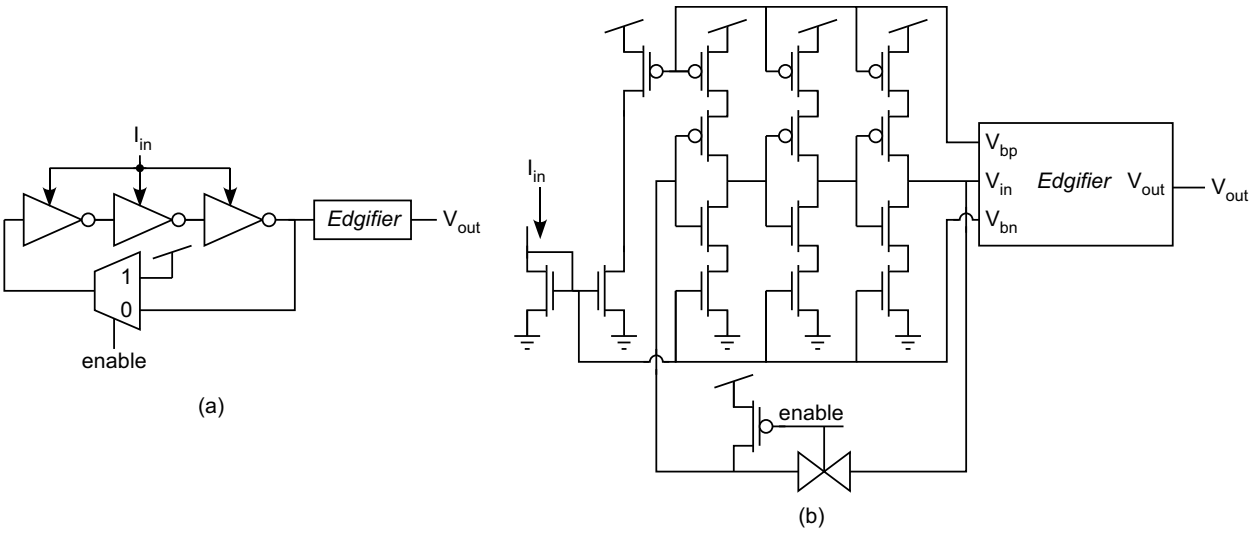

Figure 8.8: Our three-stage current-controlled oscillator with low-power edge-sharpening. (a) Block diagram. (b) Schematic.

circuit current performs no useful function and should not be allowed to dominate the power consumption.

Techniques to minimize short-circuit power dissipation include 1) equalizing the rise/fall times between the input and the output [178], which is not an option when buffering the output of an oscillator, or 2) setting the supply voltage below the sum of the threshold voltages so that the "push" and "pull" branches are not simultaneously "on" [179], which creates system-compatibility issues (e.g. multiple supply voltages and level translation) that, in some scenarios, may negate any advantages. The other way to minimize short-circuit power dissipation is to control the "push" and "pull" branches with separate non-overlapping signals. This technique is most commonly used when driving large loads—such as in clock buffers [180] or in buck converters [181]—for which it is difficult to equalize the input and output rise/fall times, and for which the consequences of short-circuit current are dire because large transistors with large current-sourcing capabilities are used. Our contribution in this oscillator is to adapt this non-overlapping gate-drive concept for use with slowly-varying

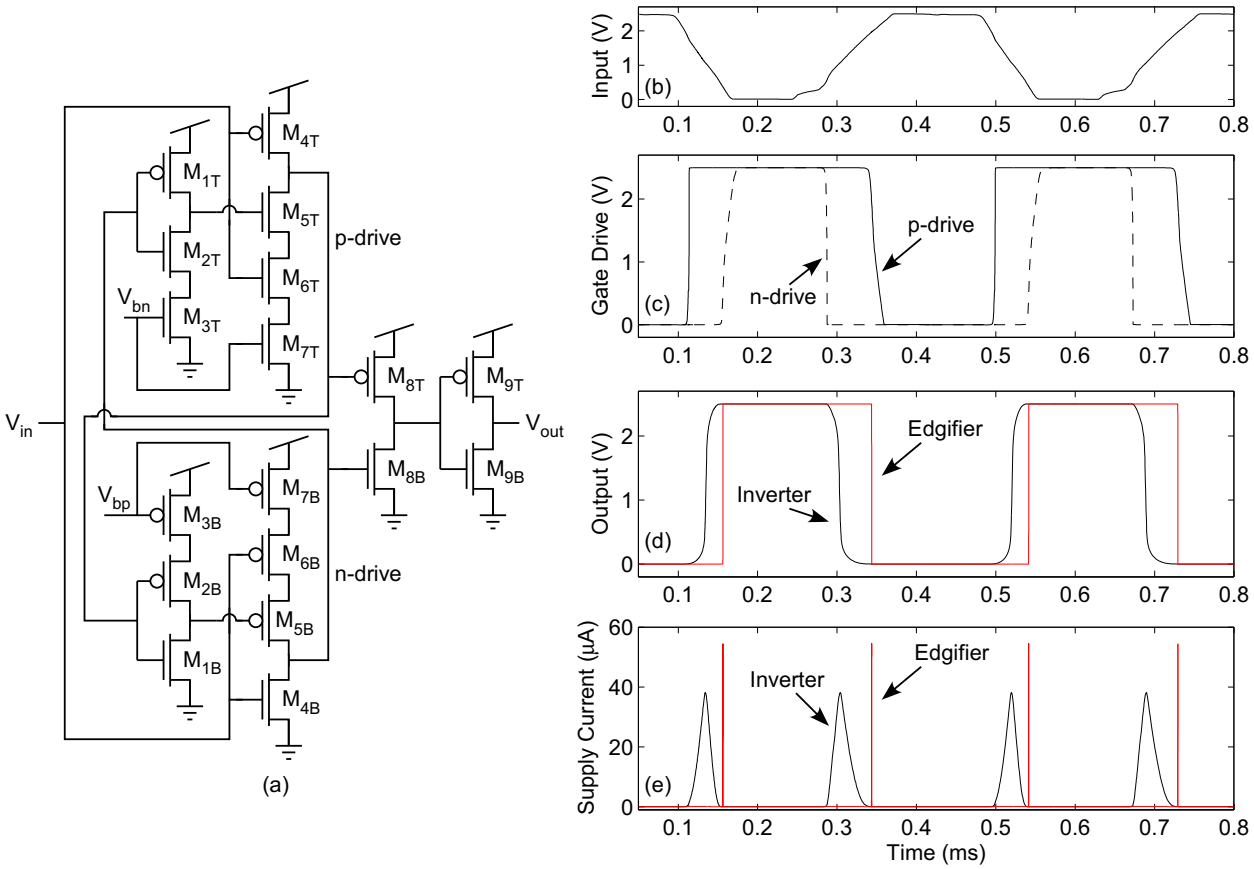

Figure 8.9: (a) Our "edgifier" circuit drives inverter $M_{8B,T}$ with non-overlapping gate signals to minimize the short-circuit current that would otherwise result from slowly rising/falling input signals. (b–e) Simulation comparing the edgifier to a single inverter. (b) Input generated by ring oscillator. (c) Non-overlapping gate drive signals generated by the edgifier. (d) Output signals of an edgifier and an inverter in response to the slowly rising/falling input in (b). (e) Supply current of an edgifier and an inverter. Non-overlapping gate drive significantly reduces the energy of each edgifier transition.

input signals. We call this the "edgifier" concept.

Our edgifier circuit, which is shown in Fig. 8.9(a), is based upon the CMOS buffer with non-overlapping gate drive presented by Yoo [180]. Yoo's circuit consists of a push/pull buffer ($M_{8T,B}$), the gates of which are driven by the logical AND of a) the input and b) the delayed and inverted version of the complementary gate signal. As a result, one transistor is always "turned off" before the other is "turned on." This technique helps to minimize the short-circuit current of a buffer for which the load is not pre-determined [180]. However, the problem that we wish to solve is the short-circuit current that is caused by slow rise/fall times at the input. The front-end gate-drive circuitry in Yoo's circuit uses standard complementary logic gates, which also suffer from the problem of short-circuit current caused by slow rise-/fall-times.

To minimize the short-circuit current that is caused by slow rise and fall times, we have added current-starving transistors $M_{3T,B}$ and $M_{7T,B}$ to the circuit's gate-drive front-end. The

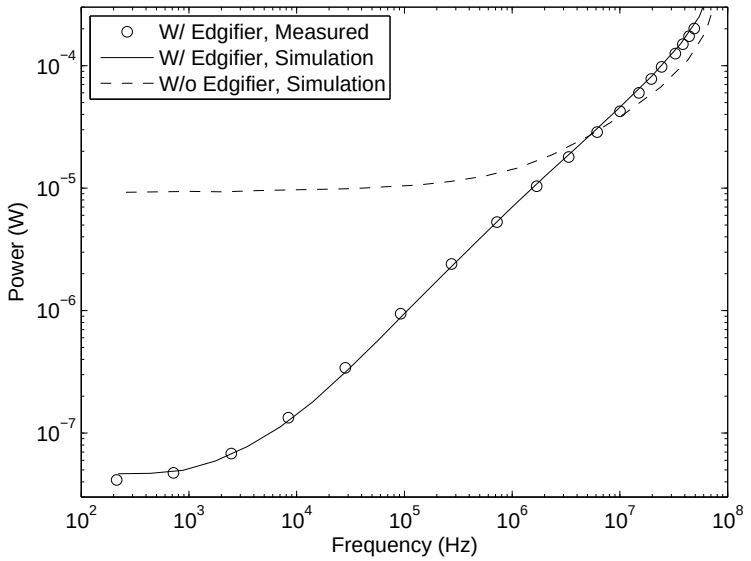

Figure 8.10: Power versus frequency of our current-controlled oscillator. The placement of an edgifier prior to any digital logic allows the power to reduce with frequency over a much larger range. The supply voltage is 2.5V.

current-starving transistors limit the short-circuit current in the front-end, while allowing inverter $M_{8T,B}$ to be driven with non-overlapping signals. To enable transistors $M_{8T,B}$ to be strongly turned off, the current-starving transistors have only been used on one side of the logic ladder.

Figure 8.9(b–e) shows simulation results that compare the operation of our edgifier to the operation of a CMOS inverter; both circuits are un-loaded. In this example, the current-starving bias in the edgifier is 1nA. We used the simulated output of the ring oscillator as a realistic input to the circuits [Fig. 8.9(b)]. Figure 8.9(c) shows the non-overlapping gate drive signals that are generated by the edgifier. The output of the edgifier is shown in Fig. 8.9(d), and is compared to the output of an inverter in response to the same input. This inverter has the same dimensions as $M_{9T,B}$. The slow transitions of the inverter output are indicative of high levels of short-circuit current. Indeed, Fig. 8.9(e) shows that the inverter draws supply current over a long duration on each transition. In comparison, the edgifier's supply current is only a short impulse.

The edgifier's reduction in the short-circuit current of succeeding logic gates can significantly reduce the overall power consumption of a circuit that contains a low frequency oscillator. This power reduction is shown in Fig. 8.10. Measured and simulated power consumption values are shown over a wide range of frequencies. The power consumption "w/ edgifier" includes the oscillator, edgifier, and one subsequent logic gate. The power consumption "w/o edgifier" includes the oscillator and one subsequent logic gate. The oscillator's rise and fall times are a constant percentage of the clock period, which results in constant power consumption for the logic gate "w/o edgifier" below 1MHz. In contrast, the power consumption of the oscillator "w/ edgifier" continues to reduce by almost three decades. Although the edgifier power consumption levels off at 1kHz in the Figure, recent simulations indicate that better sizing optimization in the gate-drive circuity can extend the line to even

lower power consumption values.

In addition to its use in low-frequency oscillators, we have found the edgifier circuit to be a useful general building block in low-frequency, continuous-time, mixed-signal circuitry. In our field-programmable analog array that is described in Chapter 10, we have used the edgifier to minimize power consumption in comparator circuits (which were the main power consumers in our earlier Hibernets 2.0 system in Chapter 9), and have also used the edgifier to serve as a translator between the low-frequency analog front-end and the digital/mixed-signal back-end. In these applications, bias currents are already generated (e.g. the bias current of a comparator), so there is no extra cost to generate the current-starving bias for the edgifier.

8.5 The Complete Charge Pump

Figure 8.11(a) shows our complete regulated charge pump, which includes all of the circuitry that was discussed earlier in this Chapter. Instead of using linear resistors in the voltage divider, we use eight diode-connected pFETs, each in their own well. Figure 8.11(c) shows measurements of the current draw of the divider branch at different voltages. The divider branch is designed to draw 100nA–1μA over the typical operating range (10V–12.5V). This current is enough to maintain stable regulation without unnecessarily wasting power. The well-to-substrate breakdown current can be seen in the top-right of Fig. 8.11(c). This breakdown occurs at a much higher voltage than the operating voltage of the circuit, so it is not a concern.

By dividing V_{out} by a factor of eight in Fig. 8.11(a), V_{out} is thus regulated to $8V_{targ}$. The measured transfer curve from V_{targ} to V_{out} is shown in Fig. 8.11(b). It can be seen that the output voltage is well-regulated from approximately 8V to 15V. Deviation at the high voltages is caused by the charge pump's limited maximum voltage $(N+1)V_{dd} = 17.5V$. In Fig. 8.11(a), error amplification is achieved by using an OTA to convert the the error into a current. Deviation at the low voltages in Fig. 8.11(b) is caused by the error-amplification OTA's bias transistor being pushed out of the saturation region.

Now that the complete details of the regulated charge pump have been elaborated, we can calculate the loop gain K that was described in Section 8.2.2. Starting from V_{out}: V_{out} is divided by $r = 8$, then converted to a current with transconductance G_m, a current mirror scales this current by a factor of 4, the current-controlled oscillator then converts this current to a frequency with a gain of $K_{RO} = 2$kHz/nA, and finally, the charge pump converts this frequency to the output voltage with a gain of K_{CP}. The total loop gain is the product of all of these components

$$K = \frac{4G_m K_{RO} K_{CP}}{r} \tag{8.6}$$

The transconductance G_m provides a way to tune the charge pump for the desired loop gain, which changes the load-regulation characteristics and the start-up time. Using K and (8.2), we can obtain the closed-loop output resistance from Table 8.1

$$R_{CL} = \frac{R_{OL}}{K} = \frac{r}{4C_p G_m K_{RO}} \frac{N}{(N+1)V_{dd} - V_{out}} \tag{8.7}$$

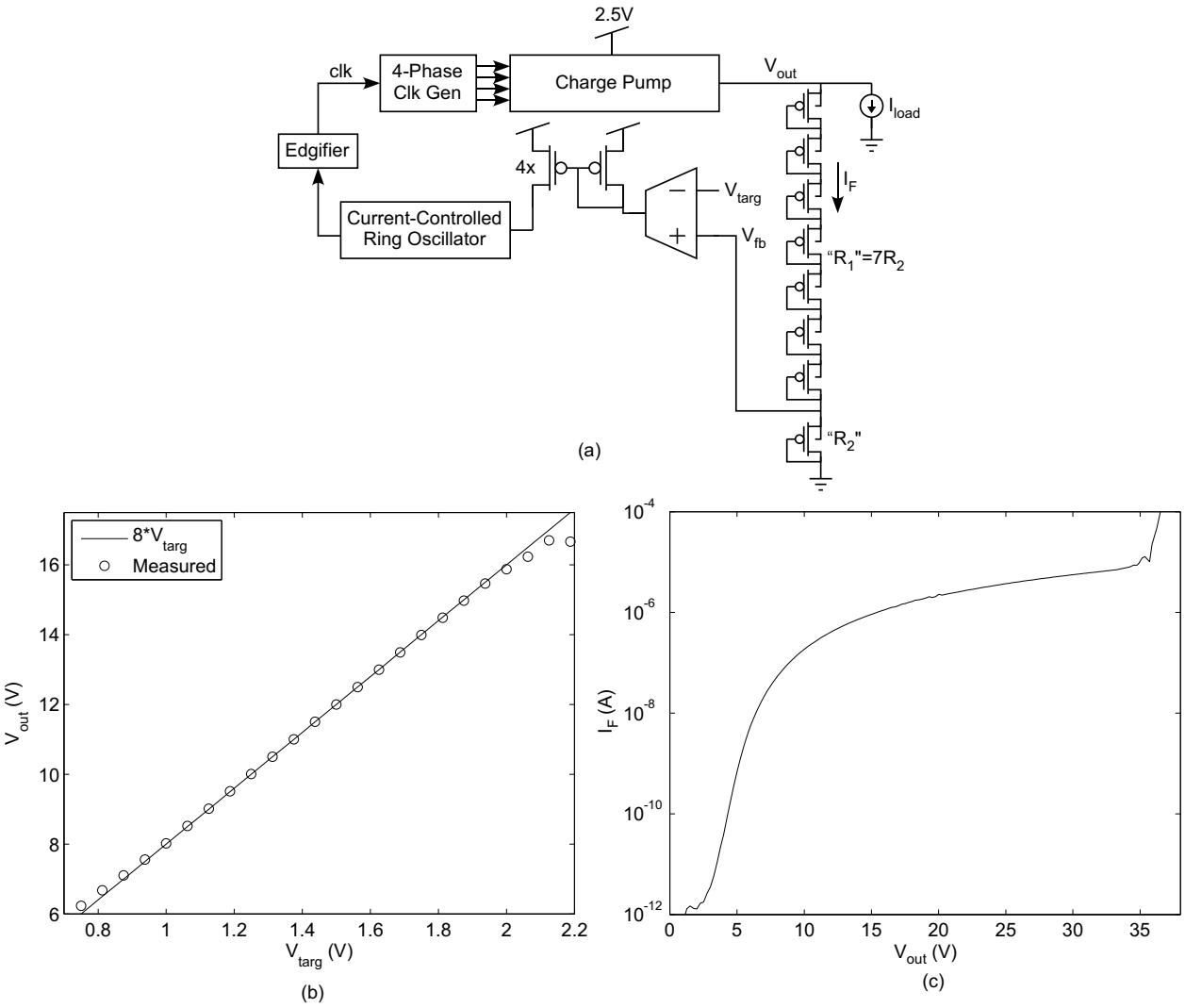

Figure 8.11: (a) Block diagram of our complete regulated charge pump. (b) Measured dc-dependence of the charge-pump output on V_{targ}. (c) Measured current–voltage sweep of the pFET-divider circuit.

To verify this expression, we have measured the open-loop and closed-loop load regulation in Fig. 8.12(a&b). The improvement afforded by regulation is clearly evident. Indeed, it would be very difficult to precisely generate an arbitrary high voltage without regulation. The output impedance is extracted from this data and is shown in Fig. 8.12(c&d). Good agreement is found between the measured results and the theoretical values for R_{OL} and R_{CL}. This agreement confirms that, when the charge pump is designed to sufficiently approach ideal characteristics, this simple analysis can be used to confidently design a high-voltage charge pump.

In a circuit, such as our charge pump, that operates beyond the rated voltage of the process, the designer should ensure that the local voltage differentials for each device are within the rated voltage range. One reason that compelled us to choose the charge pump circuit in Fig. 8.6(a) is that the use of two series switches in each stage protects the devices from any

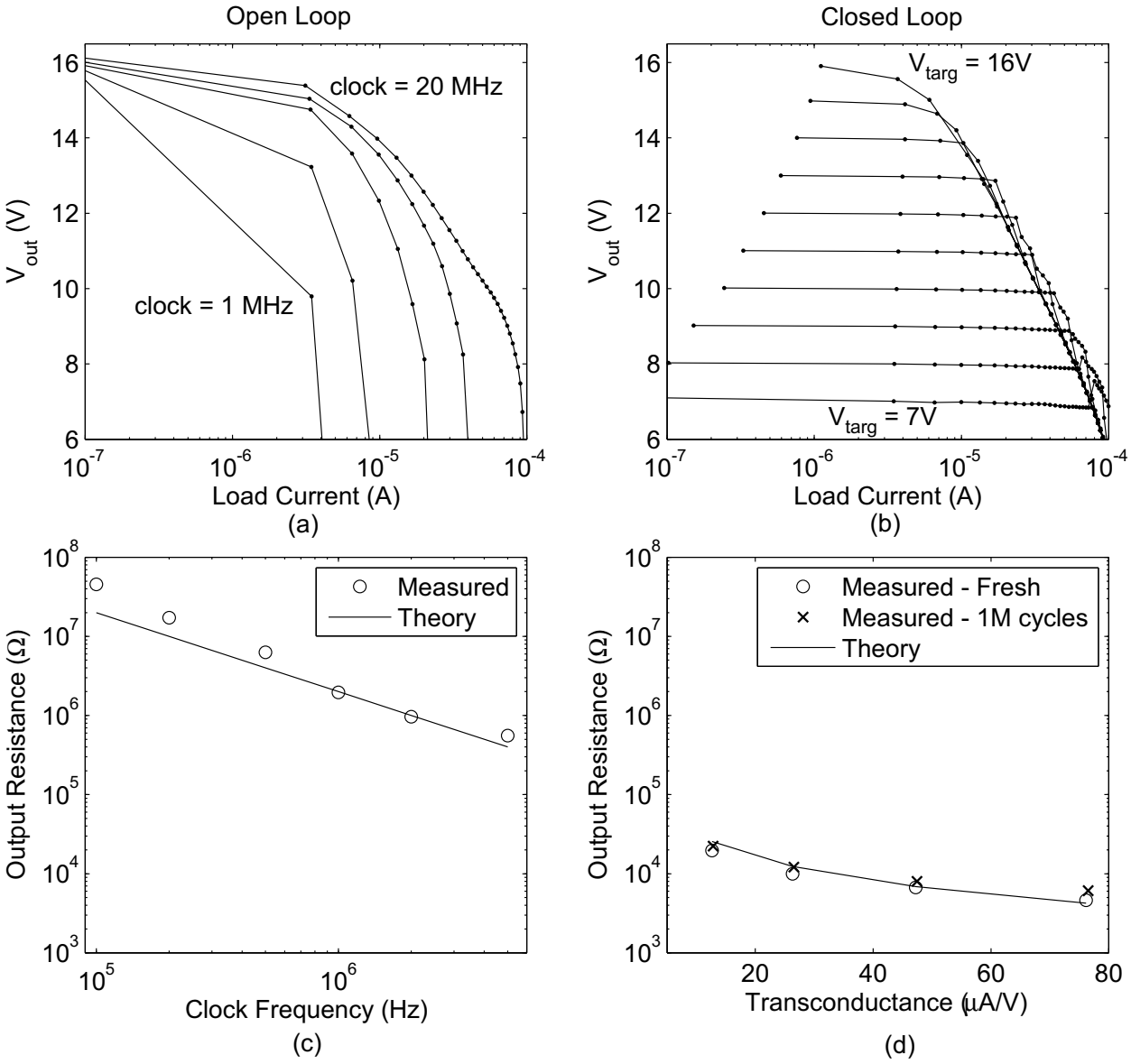

Figure 8.12: *Measured load regulation characteristics of our charge pump.* (a) Open-loop with clock frequency $\in$ [1MHz, 2MHz, 5MHz, 10MHz, 20MHz]. (b) Closed-loop with V_{targ} varied from 7V to 16V in increments of 1V. *Measured DC output impedance of our charge pump.* (c) Open-loop as a function of clock frequency. (d) Closed-loop as a function of G_m. To validate reliability, the measurement was performed with a fresh charge pump, as well with a charge pump that had previously generated 10^6 12.5V-pulses.

voltage stress greater than V_{dd} [176]. To verify that this protection ensures reliable performance under typical operating conditions, we measured the charge pump's output resistance before and after the charge pump had generated 10^6 12.5V-pulses of 1ms duration. These pulses are typical of the way the charge pump is used to program floating-gate transistors. The before-and-after measured output resistance is shown in Fig. 8.12(d). The "burned in" charge pump consistently has a slightly higher output resistance. However, the variation is

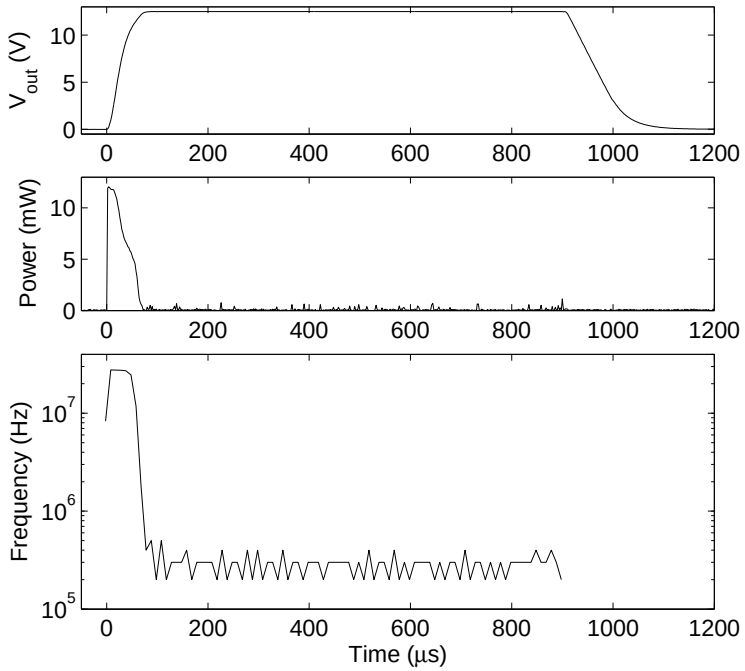

Figure 8.13: Measured transient characteristics of our closed-loop charge pump. (Top) Output voltage. (Middle) Supply current. (Bottom) Closed-loop adapted clock frequency.

small, and the number of cycles is greater than the typical rating for Flash memory, which confirms that this charge pump has great long-term reliability for our application.

The prominent characteristic of a frequency-regulated charge pump is that the frequency varies, which helps to minimize the power consumption once the target output voltage is reached. Figure 8.13 shows a measurement of the charge pump generating a 1ms, 12.5V tunneling pulse. The measured frequency over time is shown in Fig. 8.13(c). During startup, the OTA is saturated and the clock pumps at a maximum frequency of approximately 30MHz. Once the target voltage is reached, the clock is relaxed to approximately 300kHz. The resulting mitigation in supply current while the voltage is held is seen in Fig. 8.13(b). The overall energy that was used to generate this pulse was 1.45μJ.

The efficiency of a charge pump is the power delivered at the output of the charge pump divided by the total power going into the circuit. For our regulated charge pump, this input power includes the power consumed by all components, not just the charge pump. We have not emphasized efficiency because it is not a crucial specification when generating short tunneling pulses for tunneling junctions that draw a very small load current. As can be seen in Fig. 8.13, most of the energy is consumed while starting up the charge pump. However, we will briefly discuss efficiency because it is a standard comparison point for voltage converters and because it will be of interest for modifying this charge pump to generate injection-level supply voltages.

From [168], the supply current of an ideal charge pump with bottom-plate stray capacitance is

$$I_{vdd} = \left[(N+1) + \alpha \frac{N^2}{(N+1)V_{dd} - V_{out}} V_{dd}\right] I_L \tag{8.8}$$

where α is the ratio of the bottom-plate stray capacitance to the pumping capacitance, which is fixed for a given CMOS process and is in the range of 0.1–0.15 for our process [162]. The first additive term accounts for the current that is pumped toward the load. The second term accounts for the current that charges and discharges the stray capacitance. The theoretical maximum efficiency is

$$\gamma = \frac{I_L V_{out}}{I_{vdd} V_{dd}} = \frac{V_{out}}{\left[(N+1) + \alpha \frac{N^2}{(N+1)V_{dd} - V_{out}} V_{dd}\right] V_{dd}} \tag{8.9}$$

For $V_{out} = 12\text{V}$, $N = 6$, and $V_{dd} = 2.5\text{V}$, the maximum theoretical efficiency is approximately 52%. Figure 8.14 shows the measured efficiency of the open-loop and closed-loop charge pump. The regulated charge pump was measured with $V_{out} = 12\text{V}$. The charge pump achieves approximately 64% of the theoretical efficiency. Furthermore, the overhead of regulation has not significantly decreased the efficiency of the unregulated charge pump. In fact, variable-frequency regulation is able to achieve better regulation across a wider range of load currents.

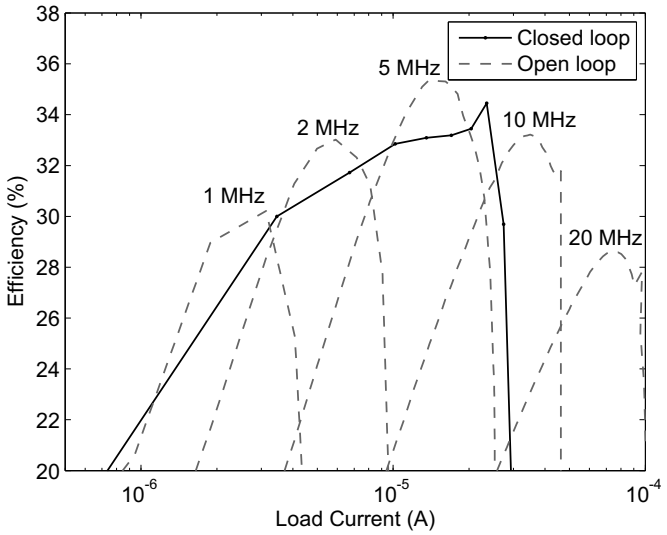

Figure 8.14: Measured efficiency of the charge pump.

Since the charge pump is designed for use in wireless sensor networks, where the supply voltage is supplied by batteries or energy harvesting and may be unstable, power-supply rejection is an important concern. Figure 8.15 shows the measured power supply rejection of the open-loop and closed-loop charge pumps. The use of regulation improves the power-supply rejection by 68dB. The regulated charge pump was measured with $V_{out} = 12\text{V}$.

Table 8.4 compares our charge pump with other high-voltage charge pumps. Aaltonen's charge pump [174] is the only regulated high-voltage charge pump that we are aware of that presents quantitative performance specifications. As a point of comparison, we have also included Li's charge pump [176], which is unregulated, but which is the circuit upon which we have based our charge pump stages [Fig. 8.6(a)]. Applying our regulation loop to that charge pump would improve many of its performance parameters by a factor of the loop gain.

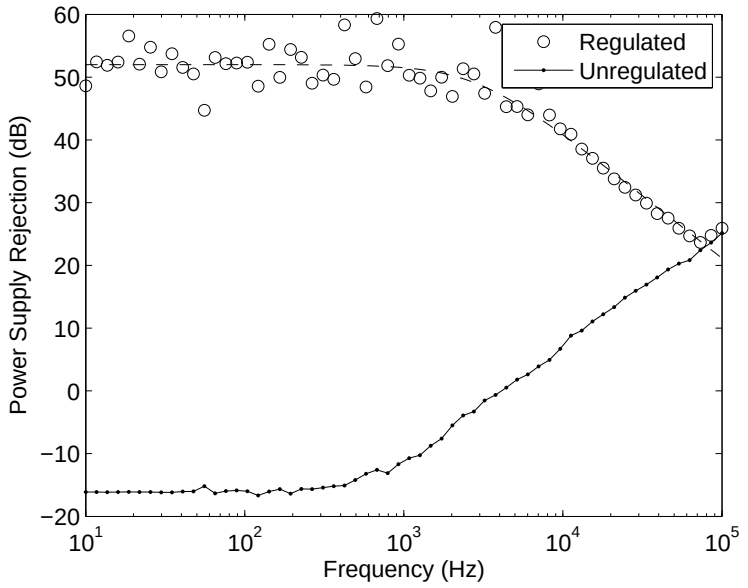

Figure 8.15: Measured power-supply rejection of our charge pump. Closed-loop regulation improves power-supply rejection by 68dB.

Table 8.4: Comparison of High-Voltage Charge Pumps

	Tech	V_{dd}	V_{out}	N	Total C	Size	R_O	γ	I_L	FOM
Ours	0.35μm	2.5V	12.5V	6	18pF	0.069mm^2	6.8kΩ	33%	25μA	4.08×10^7
Aaltonen [174]	0.35μ HV	2.5V	10V	9	14.4pF	0.14mm^2	23kΩ	18%	29μA	1.21×10^7
Li [176]	0.18μm	1.8V	7V	4	80pF	sims only	93kΩ	70%	13μA	0.05×10^7

In the Table, I_L is the load current at which the parameters are specified. R_O is the closed-loop output resistance for the first two entries, and is the open-loop output resistance for the last entry, since it was unregulated. For easy comparison, we suggest the following figure-of-merit

$$\text{FOM} = \frac{V_{out}}{V_{dd} C_{CP} R_O} \tag{8.10}$$

where C_{CP} is the total pumping capacitance. A higher FOM value is better. This figure-of-merit definition rewards large voltage step-up ratios, low output resistance, and small size—all of which are important for generating tunneling voltages. However, we should note that the comparison charge pumps were not designed for generating tunneling voltages: Aaltonenen's charge pump was designed for an electrostatic actuator for MEMS and Li's charge pump had no specific design purpose.

8.6 How to Adapt the Charge Pump to Generate the Injection Voltage

Although the charge pump described in this Chapter was designed to generate the erase (tunneling) voltage, the architecture of this regulated charge pump is also appropriate for generating the write (injection) voltage. To adapt this charge pump design for injection, we should first consider the different requirements of tunneling and injection voltages.

Tunneling requires a short pulse (<1ms) with a small load current (only the current flowing through the feedback resistors), and the voltage magnitude is large (greater than the source/drain breakdown voltage). Consequently, a tunneling charge pump has a large number of high-voltage-tolerant stages, and efficiency is not a major concern because the charge pump only supplies a small amount of current for a short time. On the other hand, the injection voltage is supplied for a longer time (>1s) with a larger load current (up to 100μA if many cells are programmed in parallel), but the voltage magnitude is moderate (less than the breakdown voltage). As a result, an injection charge pump will have fewer stages, and the efficiency is a more important specification because the injection charge pump offers the largest opportunity for reducing the total programming energy.

To design an injection charge pump for efficiency, the power-minimization routine in [168] should be used to select the number of stages and the values of the capacitors. In contrast, the tunneling charge pump in this Chapter was designed with the area-optimization routine. Essentially, the size of a charge pump can be minimized by using more stages than necessary to reduce the capacitance-per-stage. However, these excess stages reduce the efficiency of the charge pump, so this method is not desirable for an injection charge pump. Note that an area-optimized charge pump with excess stages will tend to supply $(N+1)V_{dd}$, which is greater than the desired voltage since N is greater than necessary. Regulation is thus needed to operate an area-optimized charge pump.

The output ripple is also a critical specification for an injection charge pump because it directly affects the programming accuracy. Therefore, the load capacitance must be designed to minimize this ripple. If the necessary load capacitance is impractically large, then a subsequent linear regulator can suppress the ripple.

8.7 Conclusion

In this Chapter, we have presented the design and results of a high-voltage charge pump for generating tunneling voltages. This charge pump has proven effective and reliable for erasing floating-gate transistors in our field-programmable analog array in Chapter 10. This charge pump can easily be modified to generate a supply voltage for writing floating-gate transistors.

Chapter 9

Improving the Hibernets Signal Processor

Large-scale systems of networked sensors offer a real-time understanding of the complex environments that they monitor. Since sensors must be deployed in close proximity to the phenomena of interest, often in inaccessible locations, several constraints are placed upon these sensor nodes in terms of energy efficiency and programmability. In order to be usable, they must have long lifetimes (i.e. months to years) while operating on batteries and energy-harvesting technologies. They must also be reprogrammable in the field, post-deployment, to accommodate changes in tasks.

Commercially available sensor platforms, called "motes," exist for wireless sensor networks and consist of a low-power microcontroller, a radio, and a variety of sensors [13]. These motes allow for ease of programming, even from a remote location, but they struggle to achieve the energy efficiency necessary to last for the required durations in the field. Therefore, to conserve energy, motes are often operated in a duty-cycling mode where they switch between low-power sleep states and active states, but this increases the likelihood of missing events of interest [65].

Low-power hardware can be used to detect events while the mote is in a low-power state. In Chapter 3, we introduced our Hibernets technique for sensor platforms, which complements the digital motes with a low-power analog signal processor (ASP). This ASP is capable of detecting critical events at extremely low power, and by doing so, it can selectively wake up a mote when an event of interest has been detected. In this Chapter, we present an analog signal processor (ASP) that improves upon our first Hibernets. Similar to the Hibernets 1.0 design, this Hibernets 2.0 analog signal processor (ASP) wakes a wireless sensor node when an event is detected. Events are detected by matching the quantized magnitude of the spectrum against patterns that are stored in a programmable logic array. In this ASP, we have added floating-gate-transistor-based biasing, as described in Chapter 6, which reduces the power consumption and improves the degree of programmability. We have also included the high-performance bandpass filter and magnitude detector circuits that are described in Chapters 4 and 5. Additionally, we have doubled the number of frequency channels, added multi-level quantization, and integrated the programmable logic array. All in all, the improvements made to Hibernets 2.0 have reduced the system-level power consumption from 214μW to 46.7μW, thus further illustrating the validity of analog

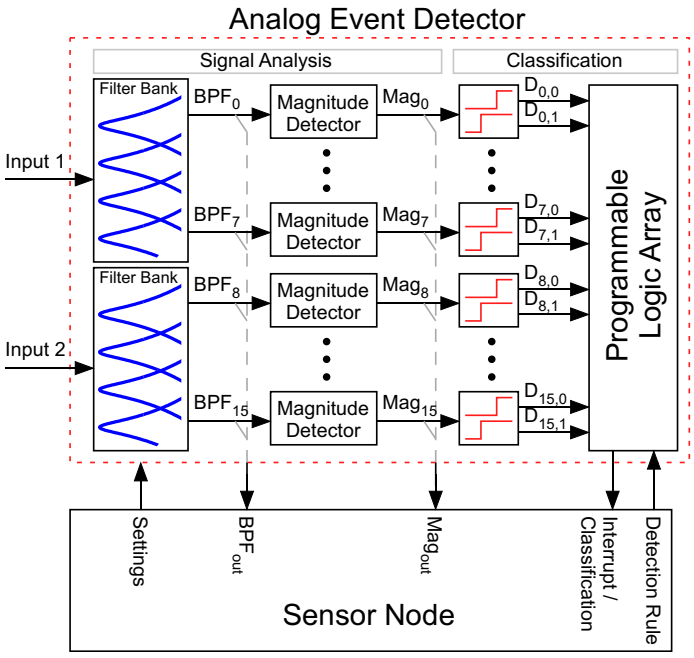

Figure 9.1: Block diagram of our analog event detector, showing how it interfaces with sensing nodes. The filter bank and subband magnitude detectors perform spectral decomposition for use by the classifier. Classification is performed with double-threshold quantization in each band followed by a programmable logic array that compares the spectral inputs with stored detection rules.

signal processing in wireless sensor networks.

The work in this Chapter was published in the Proceedings of the IEEE International Midwest Symposium on Circuits and Systems [182].

9.1 Hibernets 2.0 Architecture

Figure 9.1 shows a block diagram of our Hibernets 2.0 architecture. The analog signal processor performs event detection by comparing the spectrum of the input signal against stored templates. Spectral decomposition is performed with two 8-channel filter banks. These two 8-channel filter banks can be used separately by multiple sensors (e.g. using two microphones for directional processing) or by combining different sensing modalities like acoustic and seismic. The inputs to the filter banks can also be shared to form one 16-channel filter bank. Magnitude detectors are used in each subband to measure the signal energy. The outputs of the filters and magnitude detectors are multiplexed to output pins, which are available to the mote's analog-to-digital converter as needed. The output of each magnitude detector is connected to two compararators, allowing multi-level quantization of each band. The thirty-two digital outputs of the comparator bank are then fed into a programmable logic array (PLA) that performs template matching and generates a wake-up interrupt for the mote.

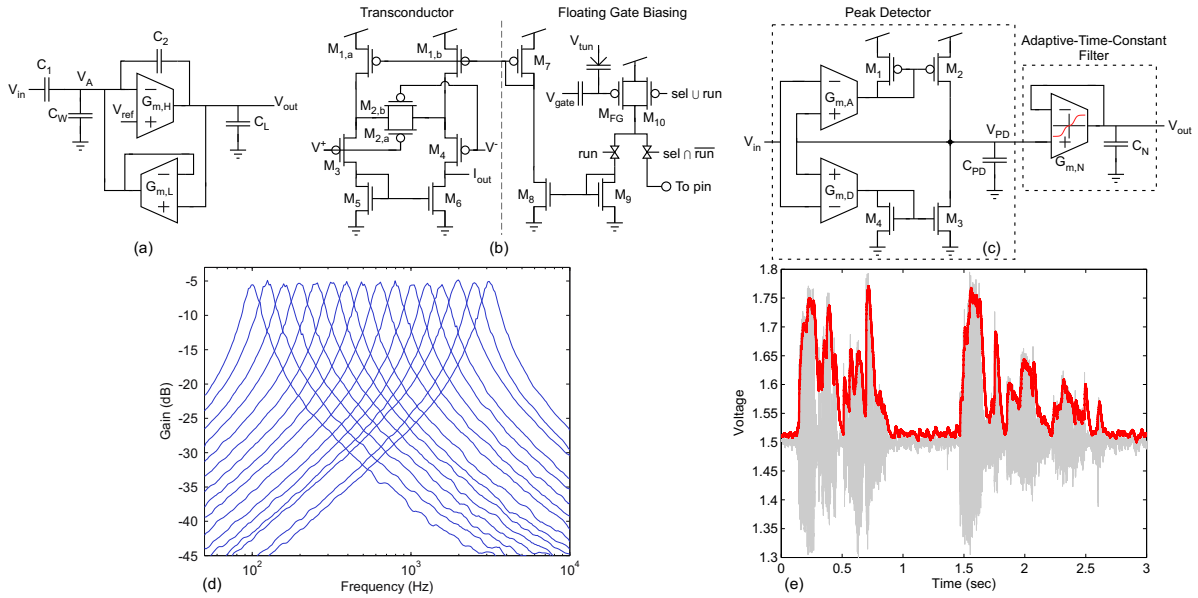

Figure 9.2: (a) The OTA-based C^4 bandpass filter circuit. (b) The linearized OTA that is used in the bandpass filter and peak detector. The bias current is set by floating-gate transistor M_{FG}, and is mirrored into the transconductor in run mode. In program mode, the drain of the selected floating-gate transistor is pinned out for programming. (c) The magnitude detector circuit. (d) Frequency response of the filter bank, which is programmed to third-octave spacing starting at 100Hz. (e) Response of the magnitude detector circuit to a speech waveform.

A two-stage uncompensated op-amp topology is used for the comparators. The PLA consists of four 32-input NAND gates that feed a 4-input NAND gate. The PLA configuration is held in SRAM, which is set via an SPI interface.

Figure 9.1 shows how the ASP is interfaced with a mote. The sensor node's digital I/O pins are used to load detection rules into the PLA and to select frequency channels. Either a digital potentiometer or the digital-to-analog converter on the mote can be used to set and adapt comparator thresholds.

9.2 Spectral Analysis Block

For Hibernets 2.0, we have used the bandpass filter and the magnitude detector that we presented in Chapters 4 and 5, respectively. These circuits are shown in Fig. 9.2(a) and (c). The bandpass filter was designed—using the design procedure in Chapter 4—for a dynamic range of 50dB and a maximum quality factor of 4.3 (corresponding to third-octave spacing).

9.2.1 Transconductor

When biased in subthreshold, the linear range of the standard differential pair is extremely limited. For increased linearity, we have used the transconductor shown in Fig. 9.2(b)

[183]. Linearization is accomplished through source degeneration by the symmetric diffusors $M_{2,a\text{–}b}$. When the width-to-length ratio (W/L) of the input pair is twice the W/L of the diffusors, this transconductor topology achieves twice the linear range of the standard differential pair.

In contrast to the symmetric bump transconductor that was used in Chapter 4, this symmetric diffusor transconductor does not achieve as high performance—the linear range is halved and the noise is somewhat higher. However, this symmetric diffusor does enable a more compact layout, as well as lower input capacitance, since its two additional transistors are half the size of the input transistors, whereas the symmetric bump adds four transistors that are the same size as the input transistors.

9.2.2 Floating-Gate Biasing

Programmable nonvolatile biases for the analog circuits are provided by floating-gate (FG) transistors [161]. As shown in Chapter 4, FGs allow the center frequency, gain, and bandwidth of each filter to be biased separately, thus enabling arbitrary frequency spacing and weighting of bands. Figure 9.2(b) shows the FG biasing circuit. The transconductor is biased by mirroring the drain current of the FG transistor M_{FG} into the bias transistors $M_{1,a\text{–}b}$.

The bias value is stored as a charge on the gate of M_{FG}. Biases are globally erased by tunneling electrons off of the FG using V_{tun}. Biases are then programmed by pinning out the drain of an individual FG transistor and applying a large source-to-drain voltage to M_{FG} in order to inject electrons onto the gate. Bias currents are typically programmed to 0.5% accuracy, though higher accuracy is achievable if necessary. Accurate measurement of the circuit parameters sets the limit to how accurately the parameters can be programmed. For this chip, all parameters are time constants.

Figure 9.2(d) shows the 16-channel filter bank biased for third-octave spacing starting at 100Hz, which is the biasing used throughout the remainder of this Chapter.

9.3 System Operation

The input signal range of the spectral analysis block was measured by sweeping the input amplitude and recording the magnitude detector output of the 1kHz band (Fig. 9.3). This experiment was done first with just the magnitude detector, and then again for the whole column, i.e. the bandpass filter and the magnitude detector combined. The Figure shows that the signal range is limited by the bandpass filter, which has a noise floor of 1.4mV_{pk} and a 1dB compression point of 157mV_{pk}. The input dynamic range, measured from the noise floor to 2.5% THD at the filter's output, is 50dB.

To verify and demonstrate the operation of the entire integrated circuit, we performed the experiment illustrated in Fig. 9.4. The input signal, shown in the top subplot, is a logarithmic chirp with pulses of tones at 200Hz and 635Hz. The full decomposition performed by the filter bank and envelope detectors is visualized in the spectrogram in the second subplot. The third and fourth subplots show the outputs of the channels corresponding to 635Hz and 200Hz, with the magnitude detector output overlayed on the bandpass filter output. To test

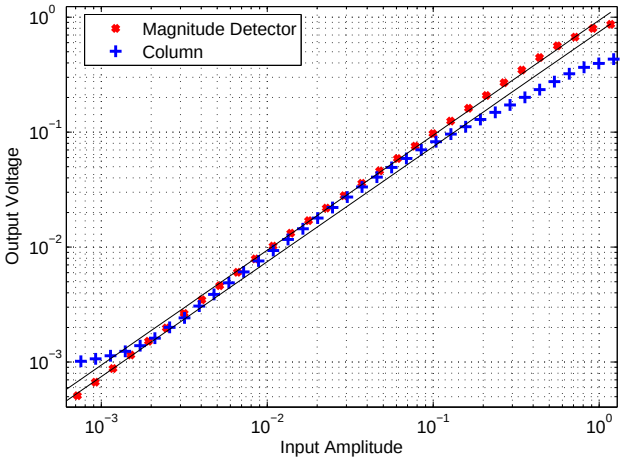

Figure 9.3: Dynamic range of the analog circuits at 1kHz. "Column" is the combined bandpass filter and magnitude detector.

Table 9.1: ASP Specifications

Specification	Value	Device	Power
Chip Area	2.25mm^2	Filter bank	1.49μW
Channels	16	Comparators & PLA	10.5μW
Chip Power	27.8μW	Read-out buffering	15.78μW
Supply Voltage	3V	External biasing	18.9μW
Technology	0.5μm	ASP + Biasing	46.7μW
Frequency Range	20Hz-100kHz	Mote (sleep)	25μW
Dynamic Range	50dB	Mote (awake)	1.5mW
		Mote (transmitting)	60mW

the detection part of the system, the PLA was programmed to perform the logical AND of channels four and nine. In the bottom subplot we see that the PLA output correctly asserts a high value when the outputs of both channels' magnitude detectors are simultaneously greater than the user-defined threshold level.

The performance of the IC is summarized in Table 9.1. All power numbers are actual measurements. The power consumption of the ASP+biasing is 46.7μW, which is better than a fourfold improvement over our previous less-integrated ASP in Chapter 3 [41], which used digital potentiometers instead of FG transistors and consumed 214μW. For the vehicle-detection scenario in the following Section, the expected lifetime (for 1500mAh battery capacity) of our analog-augmented mote is approximately 7 years. In contrast, the expected lifetime of a mote-only implementation is just 4 months.

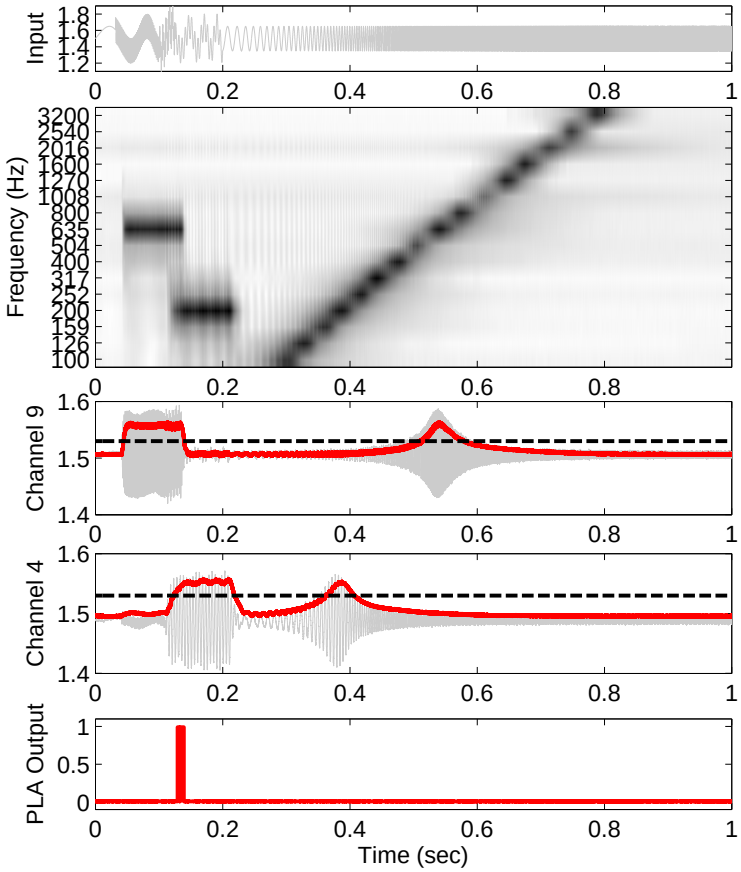

Figure 9.4: Demonstration of the event detector IC.

9.4 Vehicle-Classification Application

To evaluate the event detector in the context of a remote-sensing application, we used the same acoustic automobile-classification scenario that was used in Chapter 3. The experiment was performed using a dataset of forty 10-second audio recordings of vehicles. The objective was to detect the presence of a vehicle and classify it as either a car or a truck. Training software was implemented which used half of the recordings to learn the comparator thresholds and PLA settings that best classified an input signal as either a car, a truck, or no vehicle. The event detection IC was tested by streaming the other half of the vehicle recordings from a digital-to-analog converter into the filter bank input. Two PLA output pins were used, one to indicate vehicle presence and one to indicate vehicle type. The mote's microntroller woke up when the "vehicle presence" pin was asserted. Upon waking, the mote implemented a state machine that used the sequence of outputs from the PLA to make the final decision.

The objective of the training algorithm is to choose the front-end configuration that achieves the best detection performance. The front-end, which is shown in Fig. 9.1, has the following configurable parameters: a 132-bit logic configuration for the programmable logic array (PLA), as well as 82 analog parameters that control the center frequency and

gain/quality factor of each of the filters, the speed and tracking level of each of the magnitude detectors, and also the two quantization levels. For simplicity, we only consider the quantization levels and the logic templates; however, as we will discuss later, detection performance might be improved by incorporating more of the analog parameters into the training routine.

The training algorithm operates with a set of time-indexed training samples. All time instances of the training samples have class labels associated with them; for example, in this vehicle detection scenario, the classes are "nothing," "car," and "truck." The first step is to obtain the feature vectors—i.e. the 16-dimensional output of the ASP's spectral analyzer ($\text{Mag}_{0\text{–}15}$ in Fig. 9.1). These feature vectors are obtained by passing all training samples either through a Matlab model of the ASP or through the ASP itself; we have done so both ways and have achieved similar results.

Using an idealized ASP model[1] to obtain the feature vectors may be more realistic for most in-the-field reconfiguration scenarios, wherein a basestation uses an ideal model to train off-line and then distributes the new configuration data to in-the-field sensor nodes. However, using the the real ASP to obtain the feature vectors may have some advantages: first, the training mode will be similar to the operating mode—i.e. with the same sensor, the same ASP, and potentially the same background environment—thus enabling better training, and second, inclusion of the ASP into the training routine is an important step towards achieving in-network learning. Presently, our method of obtaining feature vectors using the real ASP is to stream the samples out of Matlab via a data-acquisition card (PCI-6259) while simultaneously reading in the ASP outputs using the same acquisition card. In principal, the mote could be used to acquire the features, and potentially perform the training routine as well.

Once we have obtained the time-indexed feature vectors, we then proceed with training. Figure 9.5 illustrates the detector training problem. Our detection architecture subdivides the feature space via the subband quantizers. These subdivisions are then labeled using PLA detection rules. Our front-end has two quantization levels: γ_1 and γ_2. To train, we sweep these quantization levels across 25 logarithmically spaced values from 20mV to 1V. For each combination of quantization levels, we find the subspace labelings that provide the best classification accuracy. This is done by labeling each region with the class that it produces most frequently; for example, if a region contains 100 instances, and 90 of those instances were for "truck," then that region is labeled as belonging to the class "truck." Based on this labeling, we then determine what percentage of instances were correctly classified. After sweeping through all values of quantization levels, we then choose the values which resulted in the highest percentage of correct classifications. Those quantization values and their corresponding detection labels are then used as our chosen configuration. Since our on-chip PLA only has room for four labels, we are limited to using only the most discriminitive labels. However, the individual input bits of the PLA can be set to *don't care* so that some labels can be combined. This combination step is done by combining labels with a small intra-class Hamming distance but a large inter-class Hamming distance.

After the training routine has finished, the configuration values are passed to the mote,

[1]The model uses ideal linear filters for the filter bank and ideal nonlinear differential equations for the magnitude detector. The model is implemented in Matlab for convenience, but could easily be implemented in a different programmaing language so that it could be performed on a basestation.

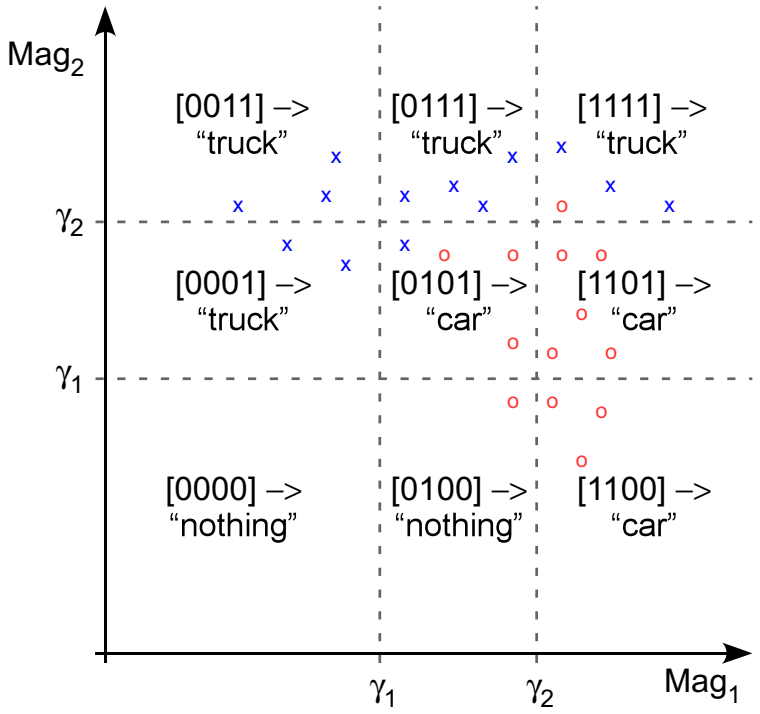

Figure 9.5: Two-dimensional illustration of the detector training problem. The goal is to find the best way to subdivide and label the feature vector space $\vec{\text{Mag}}$ using two quantization levels: γ_1 and γ_2. The o's and x's represent observations of cars and trucks, respectively.

Table 9.2: Vehicle Classification Results

	NULL	Car	Truck
Missed		20%	0%
Car	0 false alarms	70%	10%
Truck	0 false alarms	10%	90%

which sets the quantization levels using digital potentiometers and also loads the detection rules into the PLA. Otherwise, all aspects of the training algorithm are implemented in Matlab for the sake of simplicity.

The results of the vehicle classification demonstration are shown in Table 9.2. Out of twenty incidents, 80% were correctly classified and there were no false alarms in 200 seconds of noise. For the same experiment, our previous implementation in Chapter 3 [41] had two fewer misclassifications and four more false alarms. The reason that the classification results are not better than our previous system—in spite of doubling the number of channels, adding multi-level quantization, and significantly improving the circuitry and biasing precision—is due to the reduced logic capabilities of the PLA. Previously, all of the detection rules that were learned by the training algorithm were loaded into an external CPLD. However, for this implementation, only the highest-certainty detection rules could fit into the PLA. This

PLA limitation is mainly an area limitation, and increasing the PLA size would have little impact on the system power. However, transitioning to a soft classifier is expected to be a more effective use of both space and power.

As mentioned earlier, the ASP's detection performance may be improved by including more of its parameters during the training process. For example, the spectral analysis parameters can be adjusted to perform more discriminative feature extraction. First, the individual feature dimensions can be scaled by adjusting the filter gains and/or magnitude detector tracking levels, thus enabling independent control over each band's quantization levels. Second, a greater number of quantization levels can be obtained in an important frequency band by tuning multiple filters to that same frequency, but with different gains. And third, temporal information can be obtained by tuning two filters to the same frequency, but tune their magnitude detectors to operate at different speeds so that, for example, an exclusive-or of their outputs would assert upon an onset or an offset. In summary, the training algorithm does not use the full capabilities of this spectral template-matching architecture. Consequently, much higher performance may be achievable with this simple architecture. On the other hand, additional parameters would further complicate the ability to train the ASP in the field; we will consider this further in the next Section.

In addition to improving the detection performance via the training algorithm, we can also improve the performance with a better detection architecture. For example, consider a soft classifier where an array of parameterized transconductance functions is used to synthesize decision boundaries. This classifier may be trained using a boosting-type algorithm [184] that can iteratively select and train the best transconductor, and in each iteration add to the previously trained transconductors. Such a scheme may be useful for refining the decision rules in the field, since unused transconductors can be trained and added to the classifier as better information is obtained, without needing to retrain the entire classifier.

9.5 Discussion of In-Network Training

We will now consider the role of an ASP in creating adaptive sensor networks. First, we will discuss this question from an application perspective, then we will discuss the question from a front-end perspective.

9.5.1 Towards In-the-Field Training

In-the-field training can use either artificial or natural training samples. Artificial training samples are acquired off-line and might be administered to the front-end either manually or by streaming from a basestation. By administering the training samples, the network architect has control over selecting and labeling the samples.

On the other hand, natural training samples occur "naturally" within the sensor network's environment. Training with such samples is an excellent path toward learning from the environment. From an application perspective, the primary obstacle for training with natural samples is to segment/label those samples. We will first consider a few ways in which the labeling can be closely supervised. This close supervision only occurs during a "training mode," in which power consumption is not a limitation.

Labeling can be performed off-line. This approach would have an initial data-collection phase, and then would send the data to a basestation so that the events can be labeled off-line. After the data has been labeled, the remainder of training would proceed as if artificial training samples were used, i.e. train off-line and then send the detection configuration to the nodes. Note that the network would collect both the sensor data as well as the ASP's decomposition of that data, so that no ASP model would be required for training.

Ideally, the network can be deployed with little prior knowledge of the objects that it will observe. The network can then proceed to identify and label those objects, all at low power-consumption levels. The guiding principle of low-power design is that computational resources should only be used to the extent that they are needed. In our "Hibernets" approach, we have used the ASP to detect events and to direct the mote's attention to those events, thus saving power by keeping the mote turned off. If the network is to identify the initial appearance of a phenomena, and then to learn more about that phenomena, then the ASP's role will be to perform bottom-up "novelty" or "saliency" detection,[2] and to direct the mote's attention to the conspicuous time-frequency regions associated with the novel object.

In this bottom-up detection of new objects, the mote will collect features of the object and share those features with other nodes. Then the network will cluster these observations to learn how to better identify the object. Afterwards, top-down instructions can inform detection. For instance, if the object is determined to be important, then the ASP's detection configuration can be trained to identify that specific object, or the nodes can more actively share and fuse their observations about that object in order to make better decisions. Alternatively, if the object is determined to be unimportant, then the ASP can be configured to ignore that type of object in order to reduce the power that is wasted by directing the mote's attention to an unimportant object.

Additionally, the scenario described above could be a starting point for compressing data within the network. For example, visual saliency has previously been used in video compression to reduce the resolution of regions on which viewers are not likely to focus [188]. A similar scheme could be applied at various hierarchical levels within a sensor network. For instance, instead of using the ASP to make a binary decision to identify conspicuous time-frequency regions, the ASP could instead compress data by using continuous-valued "saliency measures" to adapt the sampling rate and resolution at which the mote acquires different frequency channels. At a higher level, the network could compute a distributed saliency map to determine which regions of the network have important information to share.

9.5.2 Steps to Achieve In-the-Field Training

Our Hibernets front-end is trained using a desktop computer. Let us examine how training can be done in-the-field.

The first step to achieve in-the-field training is to perform training on the mote. There are two fundamental ways to approach this step. Either the training algorithm can be performed by the mote, or the mote can be used to try different ASP settings to effectively "search"

[2]Computational saliency models have previously been presented: primarily for imaging [185, 186], but also for audio [187].

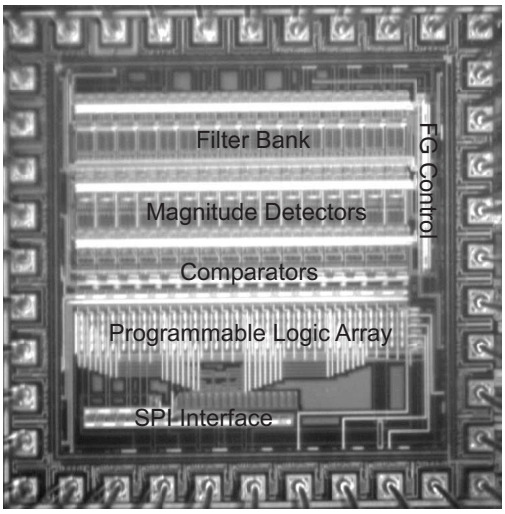

Figure 9.6: Die photograph of the 2.25mm^2 analog event detector IC.

for a good configuration.

One obstacle to porting the training algorithm to the mote is to streamline the training algorithm. There are two clear ways to streamline the algorithm. First, as the feature vectors ($\text{Mag}_{0\text{–}15}$ in Fig. 9.1) are obtained, we should reduce them to a parametric description, such as with a Gaussian mixture model. This is necessary to reduce the memory requirements of the mote. Second, we should adopt a better optimization approach than performing a brute-force sweep through the quantization levels. Assuming that these two obstacles can be overcome, training on the mote should be achievable using our current ASP and present-day higher-performance motes.

Alternately, the mote can vary the ASP's parameters until it finds parameters that yield good detection performance. For example, if the mote has a parametric description of the feature vectors, then it can generate representative feature vectors and present them directly to the ASP's classifier, bypassing the analysis block. As the mote presents these feature vectors to the ASP, it will record the classifier's output and train the classifier by altering the classifier's settings until converging on the desired detection accuracy. Such an approach may help to reduce the computational requirements of the mote.

Once training can be performed on the mote, the next step is to train on the ASP. One of the original thrusts for the analog use of floating-gate transistors was to build learning circuits [189]. Floating-gate learning circuits have struggled to find an application. However, ASP-training may be a good application for floating-gate-based learning circuits. Using such techniques, we could, for example, train the analysis block to whiten the feature vectors, or train the classifier in a manner similar to the method described in the previous paragraph.

9.6 Conclusion

We have presented a low-power programmable analog event detection IC that was fabricated in a 0.5μm process (Fig. 9.6). In comparison with state-of-the-art audio-frequency

Table 9.3: Comparison of Low-Power, Audio-Frequency Detector ICs

	Power	Application	Algorithm
This work	46.7μW	Vehicle Classification	Spectral template matching
[41]	214μW	Vehicle Classification	Spectral template matching
[22]	835nW (DSP core only)	Vehicle detection	Periodicity estimation
[190] (simulation)	I_{supply}=20μA (processing circuits only)	Glass break detection	Linear discriminator: zero-crossing rate and two energies
[191]	465μW (144μW w/o mic)	Speech detection	Phoneme-band energy modulation rate

detector ICs (Table 9.3), our ASP has the highest degree of configurability and also has one of the lowest power consumption levels. Additionally, since most audio applications use spectral analysis, our ASP's applications extend beyond mere event detection to include pre-digital signal processing. Future ASPs for resource-constrained sensing will offer higher levels of configurability, and will include more signal analysis options and better classifiers. These features will aid in developing adaptive networks.

Chapter 10

Netamorph: Simplifying the Design of Low-Power Sensor Networks with Reconfigurable Analog Circuitry

The limited power budgets of sensor networks necessitate in-network pre-processing to minimize communication costs. In the preceding chapters, we have shown that the low power consumption of analog signal processing (ASP) is well-suited for this task. However, the long development time of custom ASPs impedes their incorporation into sensor network applications. By equipping ASPs with greater reconfiguration capabilities in the form of a field-programmable analog array (FPAA), sensor network developers can more easily take advantage of ASP capabilities. Similar to reconfigurable digital processors, FPAAs allow applications developers to quickly synthesize new designs using high-abstraction-level descriptions. In this way, developers can customize ASPs to their needs and also redefine ASP operation in the field.

This chapter documents our work on designing and using FPAAs for wireless sensor networks. Section 10.1 presents our FPAA-enabled sensor node architecture. Section 10.2 presents the design evolution of our FPAA's signal path architecture. Section 10.3 presents our low-overhead reprogramming infrastructure. Section 10.4 describes how our FPAA interfaces with sensor nodes as well as how it is used. And in Section 10.5, we demonstrate our FPAA in several applications.

The work in this Chapter has been presented in a conference paper and a conference demonstration. Our first FPAA was published in the Proceedings of the International Midwest Symposium on Circuits and Systems [192]. Our first FPAA was also shown in the demonstration session at the International Conference on Information Processing in Sensor Networks [193].

10.1 A Sensor Node Architecture Incorporating Reconfigurable Analog Circuitry

Wireless sensor networks are capable of a myriad of tasks, from monitoring critical infrastructure such as bridges to monitoring a person's vital signs in biomedical applications. One

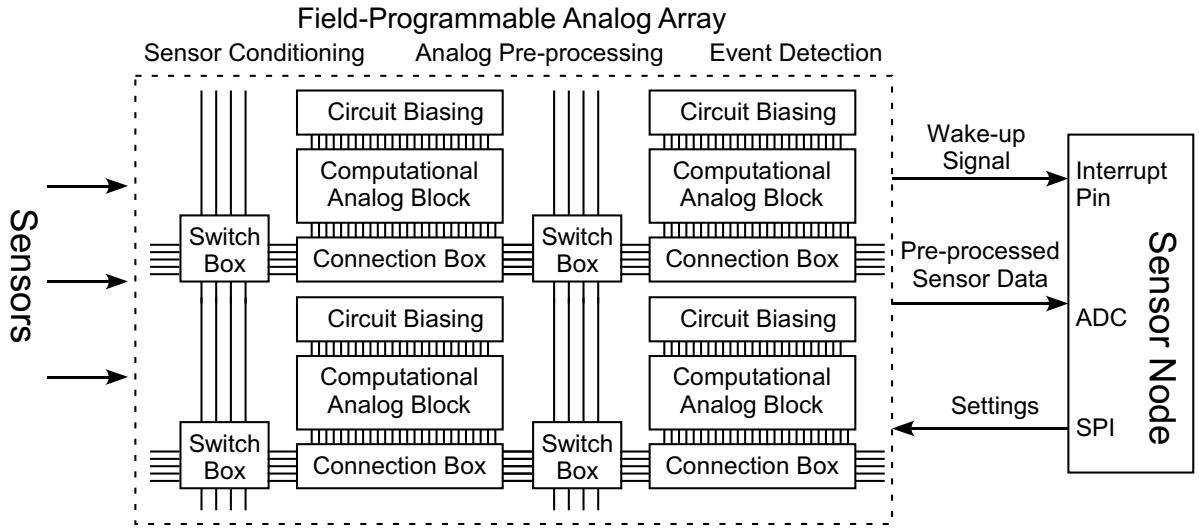

Figure 10.1: A field-programmable analog array (FPAA) used as a reconfigurable sensor interface in a wireless sensor node. The FPAA may be used for conditioning a variety of sensor outputs, to perform low-power event detection to wake up the rest of the sensor node, or to extract relevant features of the signal to reduce the bandwidth requirements of the data converter and processor.

of the primary challenges in these applications is to maximize the network's knowledge of its environment while using only the energy available within that environment. This limited energy is mostly spent on communication [7, 194], which illustrates the need for in-network pre-processing to reduce the data locally and thereby minimize this communication overhead.

In the signal path of a data acquisition system, all environmental information initially passes through the analog sensor interface, so the interface plays a critical role in efficiently collecting environmental information. A sensor interface that is optimized for the application can minimize the overall energy consumption by extracting only the necessary information—since this information is extracted early in the signal chain, minimal additional processing is required prior to transmission. For example, we demonstrated in [41] how an interface that uses analog signal processing can reduce power consumption by providing event detection for wake-up scenarios. On the other hand, a sensor interface that poorly matches the application will waste energy by capturing unneeded data, or worse, will limit the bandwidth/precision of the data so as to be unusable. Thus, an application-optimized interface is crucial for maximizing the knowledge that can be collected with the available energy. Custom analog signal processing devices are too application-specific to be a universal solution to the "efficient information collection" problem.

To enable applications developers to create interfaces that are optimized for their applications, we present a reconfigurable version of our Hibernets paradigm shown in Fig. 10.1. Here, a sensing-oriented FPAA efficiently performs sensor interfacing and information extraction. By utilizing an FPAA locally at individual nodes, an application developer can easily morph the sensor interface to the dynamic needs of the network—hence "Netamorph."

FPAAs have received increasing attention in attempts to bring the advantages of FPGAs (e.g. rapid prototyping) to traditional analog applications, such as filtering and sensor inter-

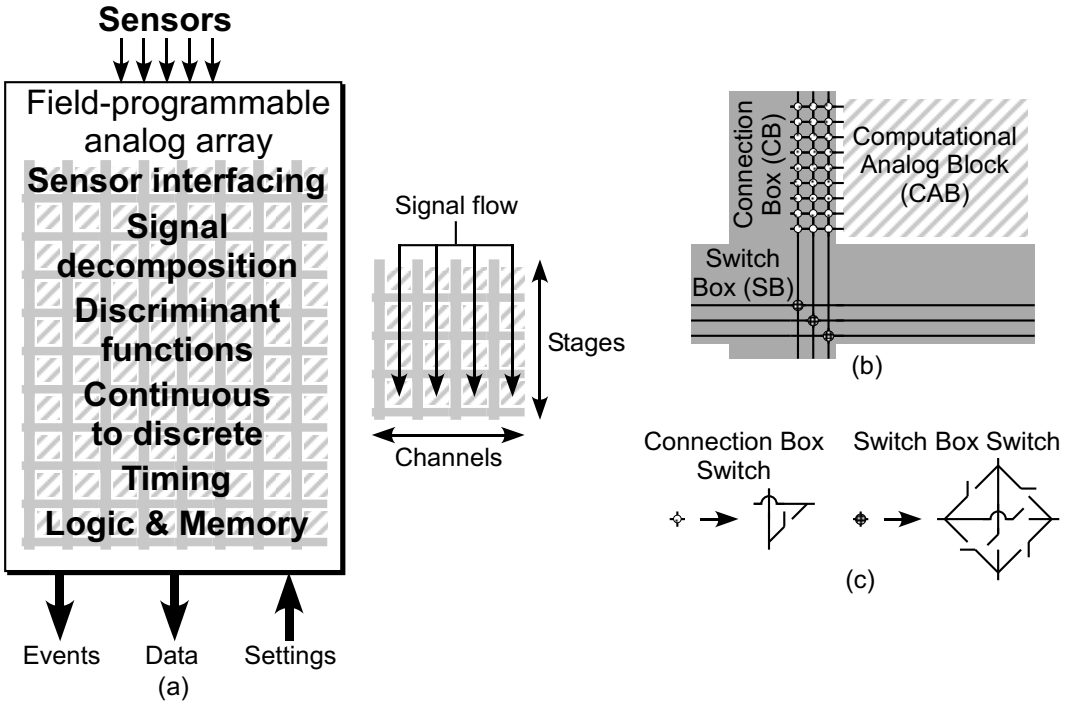

Figure 10.2: (a) Parallelized FPAA architecture for sensor networks. (b) Structure of a single computational block and its associated routing infrastructure. The computational analog block (CAB) contains a collection of circuits that are suited for the type of computation done in that stage. The connection box (CB) is used to connect the terminals of circuits within the CAB. The switch box (SB) is used to connect nodes in the CB to other CBs. (c) CB switches make a single connection between crossing lines. SB switches make any combination of six connections between four converging lines for flexible routing.

facing [195–197]. Further work has exploited the complex large-signal behavior of transistors to efficiently map signal processing and classification algorithms into analog circuits [198,199]. This potential convergence of rapid sensor-interface design with low-power signal processing makes FPAAs an enticing choice for resource-constrained sensing applications.

Prior FPAAs have not been designed for low-resource applications such as sensor networks, nor have they been streamlined to provide functionality spanning from sensor interfacing to event detection. Consequently, we have focused on developing an FPAA that meets these specific needs of sensor networks. With event detection applications in mind, we have developed a parallelized architecture, which is shown abstractly in Fig. 10.2(a). Sensor data propagates through stages from sensor interfacing through to the final extracted data, and in the process, the data morphs from a continuous-time/continuous-valued representation to a mixed-domain and then fully discrete representation. Signal decomposition is performed early in the chain, after which, data are processed in parallel channels, which allows the remaining processing to be low bandwidth and low power.

Circuits in the FPAA are located in computational analog blocks (CABs), which are similar to computational logic blocks in FPGAs, and are arranged in a grid of channels and stages as shown in Fig. 10.2(a). Each stage has an intended computational role, and

the circuits contained in the CABs reflect this role. For example, the stages devoted to discriminant functions contain a variety of voltage-to-current circuits with different shapes, which can be summed together to synthesize discriminant functions. For a given stage, the same type of CAB is included in each channel so that the same programmable functions can be implemented in parallel.

CAB circuits are connected using the switch structure shown in Fig. 10.2(b&c), which is essentially the "island-style" switch structure used in most FPGAs [200]. The terminals of CAB circuits are available in the connection box, where these terminals can be connected to other circuits using simple switches. Connections to adjacent stages and channels are made via the switch box, which contain "4-point switches" that can make connections between all pairs of converging tracks. This is known as a "disjoint" switch box, because it allows any connection within a track, but does not allow connections to other parallel tracks [200]. These "4-point switches" allow many types of connections; for example, connections can be opened to keep a net local to a block, nets can pass through the switch point using straight or right-angle connections, or two nets can pass through the same switch point (e.g. one net can pass vertically while a separate net passes horizontally). This "island-style" structure—with local connections in the connection box and higher-level connections in the switch box—enables flexible routing while reducing the parasitics caused by routing nets on long wires with many switches.

Compared to a fixed-purpose ASP, FPAAs suffer some performance degradation from routing nets through switches. Therefore, minimizing switch parasitics is one of the primary concerns of FPAA research. Figure 10.3 illustrates the parasitics in a switch matrix. If the switch is unbuffered—such as a transmission gate, a pass transistor, or a voltage-controlled resistor [201]—then the parasitics of each switch consist of four basic components. These components include 1) a switch resistance, R_{switch}, that is not zero when "on" nor infinite when "off," 2) a capacitance across the switch, C_{couple}, that causes nets in the switch matrix to interfere, 3) a capacitance to ac ground, C_{par}, that loads the nets (thus requiring more power), and 4) a leakage current, I_{par}, that limits the ability to route nets with long time constants in the switch matrix. Every net that passes through the switch matrix encounters a network of switches as the parasitics. Most of the switches are "off." If the "on" switch resistance is small enough that the net is not segmented by the resistance, and if the "off" switch resistance and coupling capacitance are sufficiently large/small to isolate the net from other lines, then the total parasitics on a net can be modeled as the parallel combination of C_{par} and I_{par}. These parasitics require extra power to achieve a given performance. Minimizing the parasitics is therefore of concern when designing FPAAs for low-power sensor nodes, and in Section 11.2 we examine how FPAA architecture choices impact the amount of parasitics encountered by a typical net.

From the above discussion, it is clear that reconfigurability adds many new optimization tradeoffs to the design of a processor. In Chapter 11, we deal with these tradeoffs in more detail—particularly in the context of wireless sensors. In the remainder of this Chapter, we describe the implementation and application of our two Netamorph FPAAs.

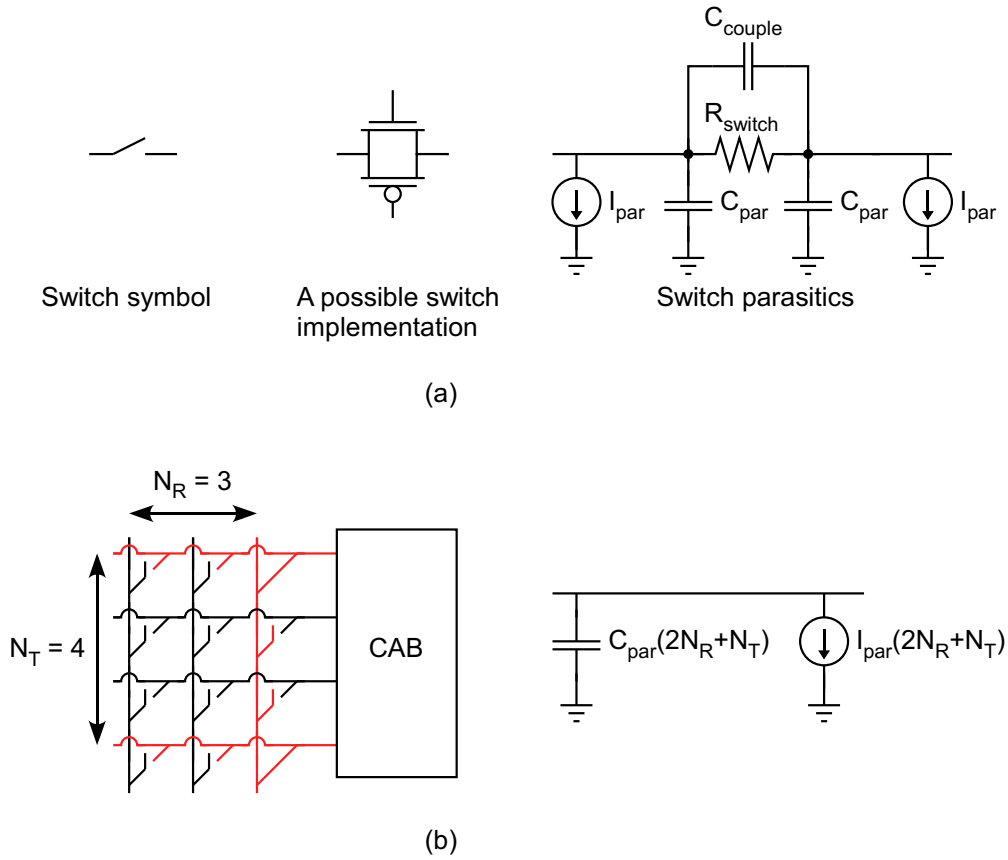

Figure 10.3: (a) Whether a switch is implemented with a complementary transmission gate (as shown), a pass transistor, or voltage-controlled resistors [201], the parasitics can be modeled the same. (b) For a net that is routed in a switch matrix, most of the switches connected to the line are "off." If R_{switch} is small enough for "on" switches and large enough for "off" switches, then the parasitics for the line can be modeled as the parallel combination of the capacitance and leakage to ground.

10.2 Parallelized FPAA Architecture for Embedded Signal Processing

Previous FPAA designs have demonstrated the ability to synthesize complex analog circuitry [198]; our goal is to establish their viability in embedded systems such as wireless sensor networks. Specifically, we would like to establish the use of FPAAs within energy-constrained systems that monitor phenomena such as audio, vibration, and motion, which have sufficiently high bandwidths (>100Hz) to challenge the throughput of typical wireless sensor networks.

We have developed two versions of FPAAs which have evolved out of our Hibernets designs. Netamorph 1.0 was small, with just 4 CABs, and its purpose was to resolve the details of the reconfigurable infrastructure before scaling up to a larger FPAA. Netamorph 2.0 has been scaled up to 80 CABs, has a higher level of integration, and has better software tools for creating FPAA designs, all of which make this second version very usable. Both

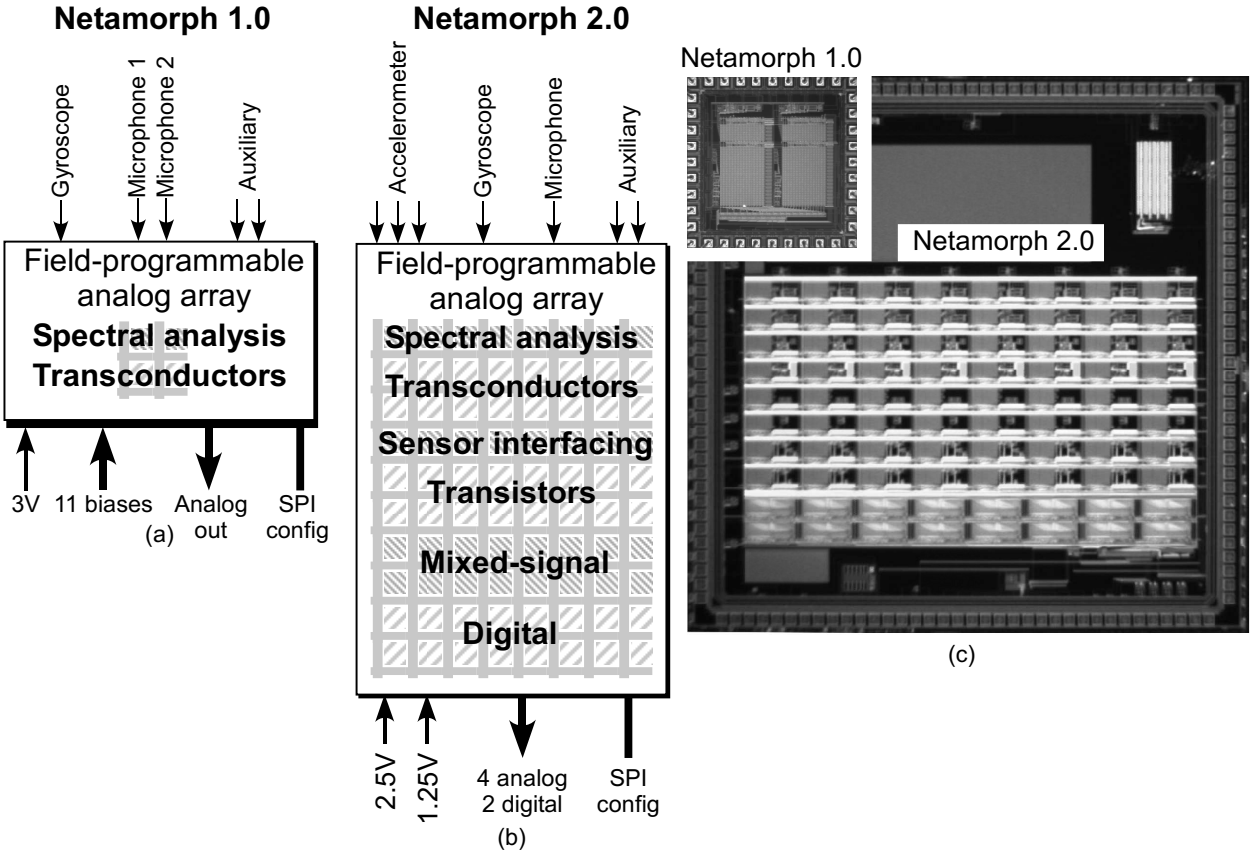

Figure 10.4: (a) Processing architecture of Netamorph 1.0. Two spectral analysis CABs are followed by two general purpose CABs for subband processing. (b) Processing architecture of Netamorph 2.0. The CABs are arranged into eight parallel channels, each with ten stages. (c) Die photos showing the relative size of the two generations of FPAAs.

versions use SRAM-controlled switches; more details on the switches are provided in Section 11.3.

10.2.1 Netamorph 1.0

Netamorph 1.0 was made in a standard 0.5μm CMOS process and was 2.25mm^2. An abstract view of the processing architecture is shown in Fig. 10.4(a) and the die photo is shown in Fig. 10.4(c). This FPAA consists of four computational analog blocks (CABs), including two for spectral analysis and two for subband processing. The spectral-analysis CABs include filters and magnitude detectors. These circuits all have tunable biases, allowing the user to perform several common types of analysis on a range of signals before further processing. Further processing takes place in the subband processing CABs, which provide access to elements of a smaller granularity, such as OTAs, capacitors, and individual transistors. The computational elements are listed in Table 10.1. Demonstrations are given in Section 10.5.

Table 10.1: Computational Elements in Netamorph 1.0

2 C^4's	2 envelope detectors	2 LPFs
16 FETs	8 capacitors	10 OTAs

Reconfiguration is achieved via programmable switches in the connection box (which is used for intra-CAB routing) and the switch box (which is used for inter-CAB routing). The connection box consists of a crossbar configuration for flexible local routing. To facilitate ease of integration with a sensor node, we implemented these switches using SRAM-controlled transmission gates. Each switch had an SRAM memory cell that set it to "on" or "off." To load values, we used a row-by-row method that would load the state of all 16 switches in a given row. The configuration was written into the SRAM array using an on-chip serial peripheral interface (SPI). In total, the FPAA had 1436 switches, with the potential to route 40 unique nets. Netamorph 1.0 had no internal biasing so an external DAC was used.

Some of the ways in which Netamorph 1.0 influenced the design of Netamorph 2.0 are described below.

1. *Switch box.* In the switch box, we implemented a variety of connection types (such as crossbar, crossover, and four-way switch points) to evaluate their value within sensor networks. We found that the four-way switches allow the best use of routing lines, so Netamorph 2.0 primarily uses four-way switches.

2. *Number of routing lines.* The connection boxes had 14 routing lines that were available for connecting terminals of CAB circuits (excluding lines dedicated for ground and V_{dd}). The "Transconductors" CAB had 34 terminals while the "Spectral analysis" CAB only had 9 terminals. We found that 14 routing lines was too few for a CAB with 34 terminals, while it was excessive for a CAB with 9 terminals. This influenced two design decisions for Netamorph 2.0. First, all CABs should have the same number of terminals to avoid a mismatch between the number of terminals and the number of routing lines. Second, a connection box should have at least one routing line for every two terminals (we used 9 routing lines for 16 terminals in Netamorph 2.0).

3. *Global connections.* Netamorph 1.0 contained a large number of global connections: ground, V_{dd}, two global lines, and four global channel lines. The global supply nets (ground and V_{dd}) were useful. But the rest of the global connections were excessive. In Netamorph 2.0, we used one global line for a midrail reference, as well as one global stage/channel line in each stage/channel.

10.2.2 Netamorph 2.0

Netamorph 2.0, shown in Fig. 10.4(b), was fabricated in a standard 0.35μm CMOS process and was 25mm^2. This FPAA consists of 80 computational blocks arranged in an 8-channel by 10-stage signal flow. A crossbar switch matrix connects the 16 terminals of the computational elements at the block level, and a switch box connects 6 routing tracks between neighboring blocks. Additionally, a long track in each stage/channel routes signals

globally. All 20,380 switches are implemented using a transmission gate controlled by a local SRAM bit.

We have included high-granularity computational elements in the FPAA to reduce the number of switches in the signal path. Our FPAA's computational elements are listed in Table 10.2; these elements represent a variety of granularity and function to improve performance and application fit. The elements are applicable to most signal-processing applications and are grouped by function into stages to streamline audio and vibration applications. These groups are described below.

Table 10.2: Computational Elements in Netamorph 2.0

8 C^4's	56 OTAs	8 inverters	16 envelope detectors
8 LPFs	8 multipliers	32 comparators	48 current sources/sinks
56 caps	8 op-amps	8 bump circuits	16 pulse generators
8 PNPs	16 resistors	8 time-to-voltage	16 asymmetric integrators
16 S/Hs	144 FETs	32 JK flip flops	16 6-input 2-output LUTs

1. *Spectral analysis*: Contains programmable filters, envelope detectors, and OTAs with reconfigurable bias terminals to synthesize frequency decomposition algorithms.

2. *Transconductors*: Contains a variety of linear and nonlinear transconductance elements for synthesizing G_m–C networks and discriminant functions.

3. *Sensor interfacing*: Contains op-amps and resistors to build reconfigurable sensor interfaces.

4. *Transistors*: Used to synthesize computational elements that are too specialized to include as dedicated elements.

5. *Mixed-signal*: Contains comparators, S/Hs, programmable-width pulse generators, etc. Designed with current-starved and non-overlapping push/pull logic to nullify short-circuit current caused by interfacing with slow signals from the preceding stages.

6. *Digital*: Contains flip flops and lookup tables. Used to add digital control to analog circuits and to generate event-detection and data-ready interrupts for the application processor. Connected to the FPAA's SPI pins for run-time writing/reading of synthesized registers.

10.3 Memory Programming

The benefits of reconfigurable sensor interfacing and reconfigurable analog signal processing makes FPAAs enticing for resource-constrained sensing applications. However, the cost of dense analog parameter storage—either high quiescent power to refresh volatile storage or high infrastructure overhead to write nonvolatile storage—limits the use of large-scale FPAAs in low-power systems. To make large-scale reprogrammable analog circuits a reality

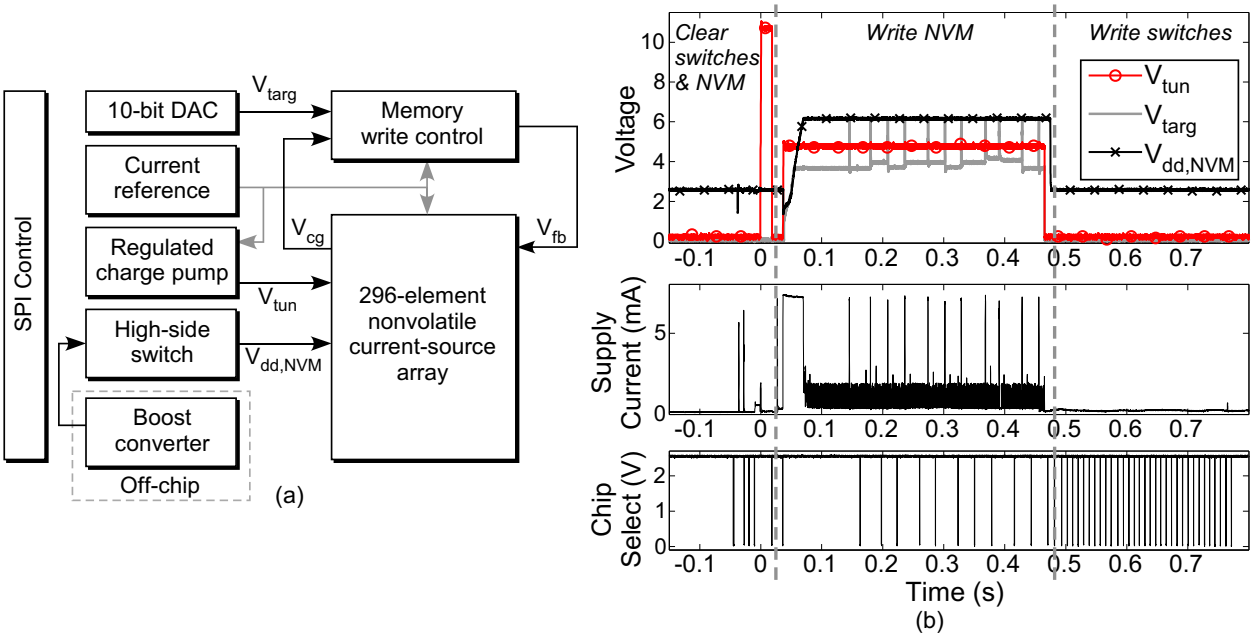

Figure 10.5: (a) Block diagram of the FPAA's analog nonvolatile memory (NVM) programming system. (b) Measurement of the process of loading a design into Netamorph 2.0. The supply current is measured flowing into the FPAA board. The SPI "chip select" pin indicates data transfers from the application processor.

for low-power sensing systems, we have developed a low-overhead, highly-integrated FPAA-programming infrastructure. This programming infrastructure is used in Netamorph 2.0. Only 1–48mJ is required to reprogram, depending on the design which is loaded into the FPAA, thus enabling Netamorph 2.0 to be used in low-resource systems.

10.3.1 Memory Programming Infrastructure

The programmable characteristics of Netamorph 2.0's computational elements—e.g. time constants, G_m's, pulse widths—are controlled by 296 analog nonvolatile memory (NVM) elements. These memory elements consist of programmable current sources that are based upon floating-gate transistors, as described in Chapter 6. Floating-gate transistors have no resistive connection to their gate; instead, a "control gate" couples capacitively onto the transistor's "floating gate." As a result, the floating-gate charge, which can be modified using Fowler-Nordheim tunneling and hot-electron injection, creates a programmable and nonvolatile threshold-voltage shift from the perspective of the control gate.

We have developed a highly integrated, energy-optimized system—Fig. 10.5(a)—for programming analog values onto floating gates. The NVM elements and write control circuit are extensions of our continuous-time floating-gate programmer in Chapter 6, and are shown in detail in Fig. 10.6. The reprogramming process is shown in Fig. 10.5(b) and is described below.

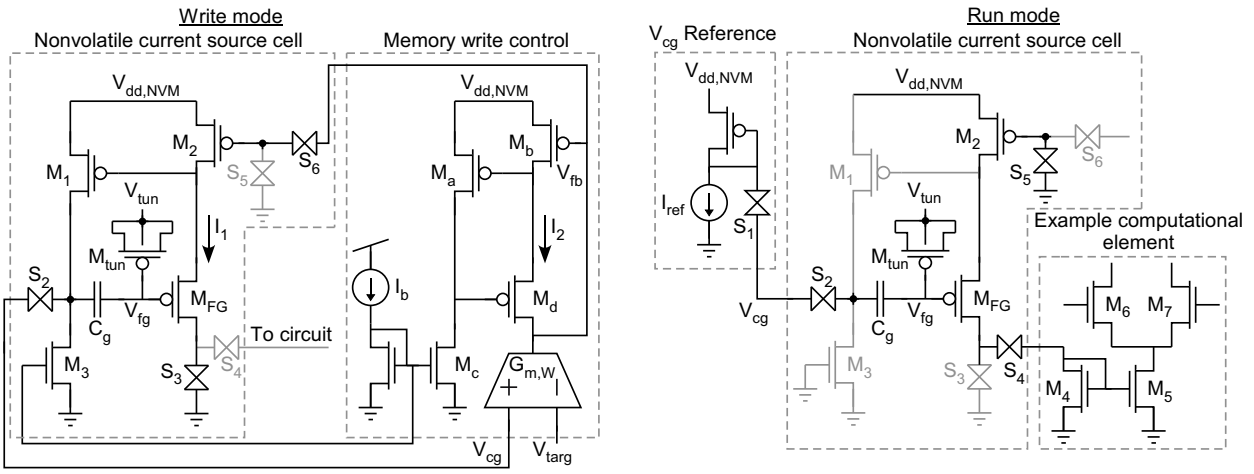

Figure 10.6: Detail of the NVM cell. In write mode, a local feedback loop ($M_{1,3}$) around the floating-gate transistor M_{FG} linearizes the exponential injection characteristics. NVM cells are individually connected to a write control circuit which modifies the current injected onto V_{fg} (by modifying I_1) until the charge on V_{fg} causes V_{cg} to match V_{targ}. The regulated-cascode current mirror, $M_{a\text{–}d}$, replicates the memory cell structure to improve matching between I_1 and I_2. In run mode, each NVM cell is connected as a current source to its respective computational element.

10.3.1.1 Clear Switches & NVM

First, the FPAA is reset by clearing the SRAM that controls the switches and by block erasing the analog floating-gate memory. The SRAM is cleared globally with a reset command. As described in Chapter 11.3, a global clear significantly reduces the overhead of SPI transfers. Block erasure of the analog memory is performed by applying a 10.5V pulse on V_{tun} to tunnel electrons off of all floating gates. This voltage pulse is generated by an on-chip high-voltage charge pump.

10.3.1.2 Write NVM

Next, analog values are written to the NVM. Writing is performed by first raising the memory supply voltage, $V_{dd,NVM}$, to 6V to facilitate injection, and then by sequentially connecting individual elements to the write-control circuit to store V_{targ} into the NVM. V_{targ} is set to the desired value for each element by an on-chip DAC. During the write process, V_{tun} is set to 4.5V using the on-chip regulated charge pump to avoid reverse tunneling through M_{tun}.

10.3.1.3 Write Switches

Finally, switches are set so as to wire the FPAA for the desired functionality. We have written a PC program to translate a user-generated netlist into a list of switches to be programmed.

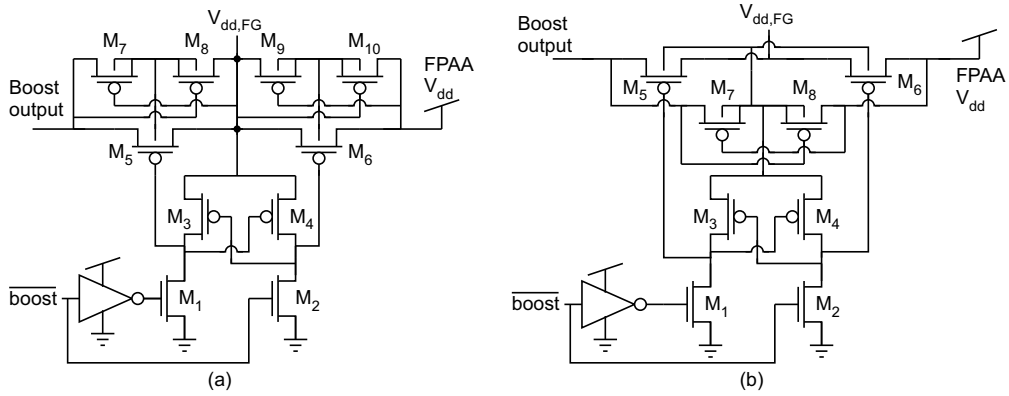

Figure 10.7: (a) Schematic of the high-side switch developed for the NVM programming system in 10.5. (b) An improvement upon the switch.

10.3.2 High-Side Switch

In run mode, the floating-gate supply voltage $V_{\mathrm{dd,FG}}$ should have a low-impedance connection to the chip supply voltage. During programming, $V_{\mathrm{dd,FG}}$ is raised to a higher voltage by the off-chip boost converter. Making these connections requires a high-side switch (i.e. the connection is made at the positive supply rail) that operates at varying voltages beyond the battery voltage. Demand for such switches is increasing as electronics include more sophisticated power management—that is, multiple supply voltages receiving power from multiple sources (such as batteries and energy harvesting) and aggressive gating of unused subsystems.

Such switches are typically nFET-based because the lower mobility for electrons compared to holes results in a better tradeoff between "on" resistances and device size. However, since the gate voltage that is required to turn on the nFET will be higher than the circuit's voltages, a charge pump is required to generate a higher gate voltage. In low-current applications, the area cost of a charge pump will negate the advantages of lower mobilities in nFETs. The other advantage of nFETs is that the bulk terminal is easier to bias since it need only be less than the voltages that are switched. In contrast, pFETs require dynamic biasing of the bulk terminals since they must always connect to the highest voltage. Our high-side switch simplifies the dynamic biasing of the pFET bulk terminals.

For our memory programming system, we have developed a simple high-side switch shown in Fig. 10.7(a) that has broad applications. Transistors M_5 and M_6 are the switches that connect $V_{\mathrm{dd,FG}}$ either to the boost converter or the FPAA V_{dd}. Transistors $M_{1\text{–}4}$ form a standard level shifter, which shifts the selection signal $\overline{\mathrm{boost}}$ and its complement to the necessary voltage to turn off the unselected switch (e.g. turn off M_5 if $V_{\mathrm{dd,FG}}$ is connected to FPAA V_{dd}). Additionally, well-selection transistors $M_{7\text{–}8}$ and $M_{9\text{–}10}$ bias the wells of the switch transistors at the highest voltage so that the source/drain junctions do not forward bias.

The switch in Fig. 10.7(a) has one disadvantage: it is only able to connect $V_{dd,FG}$ to the higher voltage. This is because the selection signal is level shifted to $V_{\mathrm{dd,FG}}$, and if $V_{\mathrm{dd,FG}}$ is connected to the lower voltage, then the selection signal is not level-shifted high enough

to turn off the opposite switch. In our programming system it is never necessary to connect to the lower voltage, but to improve the switch for a wider range of applications we present the improved switch in Fig. 10.7(b). Here, a single well-selection transistor pair M_{7-8} biases the wells of all pFETs, as well as the voltage to which the selection signals are level-shifted, to the highest voltage in the circuit. This ensures that the deselected switch can always be turned off.

In summary, by designing appropriate gate- and bulk-biasing circuitry, pFETs are easy to use as high-side switches in low-current applications. Furthermore, because of the high breakdown voltage of the pFET's well-to-substrate junction, which is approximately 35V in 0.35μm, our switch design can be extended to safely pass voltages far in excess of what an nFET switch can handle.

10.3.3 Summary of FPAA Programming

Analog memory writes are controlled by an on-chip feedback loop, and the only interaction from the application processor is to specify the memory address and DAC code-word. As a result, we incur significantly less overhead than systems which require a processor in the feedback loop [202]. The entire process in Fig. 10.5(b) consumed 2.35mJ, 33% of which is consumed while starting the external boost converter and can be largely eliminated by generating $V_{dd,NVM}$ with an on-chip charge pump (similar to our V_{tun} charge pump).

In Fig. 10.5(b), 10 NVM and 33 switches were written. The energy breakdown is approximately 0.12mJ per NVM, 6.4μJ per switch, and a constant 0.94mJ each reconfiguration cycle. These data are used to estimate the energy to load larger designs into the FPAA. For example, the design in Fig. 10.16 uses 478 switches, 29 LUT bits, and 52 analog NVM, at a total reconfiguration energy of 10.42mJ.

10.4 Using the FPAA

10.4.1 Interface PCBs

Our end goal with this FPAA work is to integrate the Netamorph FPAAs into a wireless sensor node in a way that enables us to easily monitor a range of phenomena. To that end, we created a printed circuit board (PCB), as shown in Fig. 10.8, for each FPAA version.

10.4.1.1 Netamorph 1.0 Interface PCB

The PCB, shown in Fig. 10.8(a), includes a variety of sensors, two FPAAs to enable scalability, a TelosB mote connector, a digital-to-analog converter (DAC) for providing bias voltages, as well as a complex programmable logic device (CPLD). For the sensors, we chose to focus on the relatively high-frequency phenomena that are traditionally very taxing on a wireless sensor network's power budget, including motion, audio signals, and various forms of simple harmonic motion. To monitor these phenomena, we equipped the board with a gyroscope, two microphones placed at opposite ends of the board to enable directional sensing, and a mini-stereo port to enable future expansion.

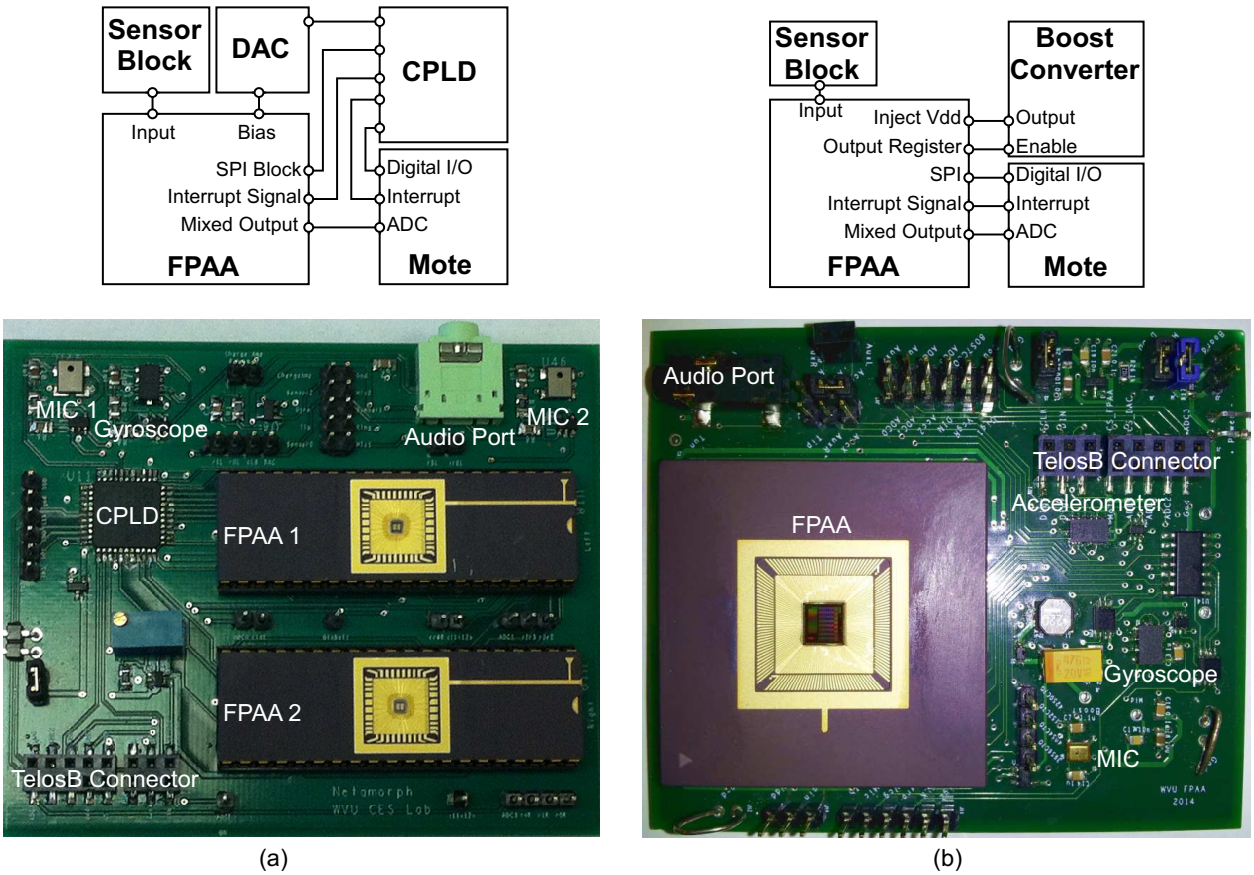

Figure 10.8: (a) PCB for interfacing Netamorph 1.0 to a wireless sensor node. This board incorporates a variety of sensors, a CPLD, a DAC (on the underside of the PCB), and a socket for connecting a TelosB mote. The PCB measures 3.4" x 2.8". (b) PCB for interfacing Netamorph 2.0 to a wireless sensor node. As a result of the higher level of integration in version 2.0, the DACs and the CPLD are no longer included. The PCB measures 3.1" x 2.6"

By including two FPAAs on the same board, we were able to scale up our architecture and build more sophisticated ASP designs. The FPAAs include SPI blocks that can be programmed directly through the attached TelosB mote's general purpose input/output (GPIO) pins. The on-board CPLD was used to minimize the number of TelosB pins that were used for digital I/O, thus freeing up more pins to be configured as ADCs. Additionally, the CPLD was used to define even more complex wake-up events. We also simplified setting individual bias points for all of the ASP blocks by including DACs which can be set and adjusted directly by the mote.

10.4.1.2 Netamorph 2.0 Interface PCB

The PCB, shown in Fig. 10.8(b), includes the FPAA, a variety of sensors, a shift register for enabling/disabling sensors, a boost converter for programming nonvolatile memory, a current reference for temperature compensation, 2.5V regulators for analog and digital supplies, a 1.25V reference, and comprehensive power probing.

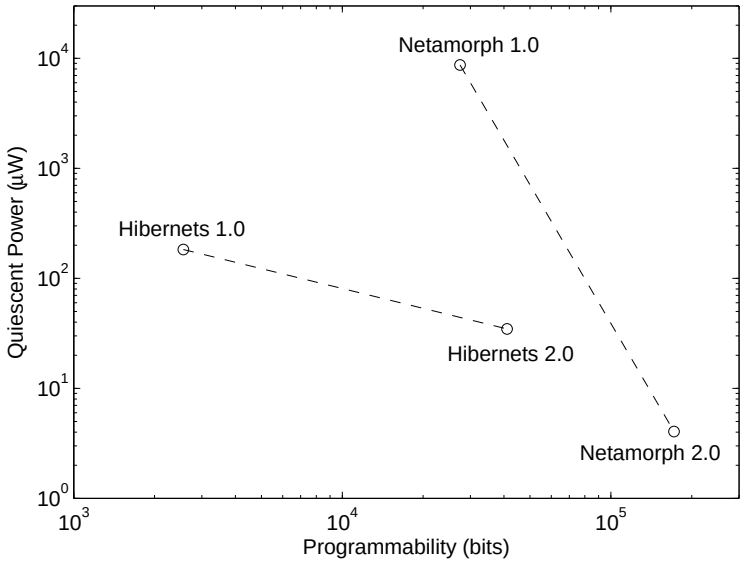

Figure 10.9: Comparison of the level of programmability and the "quiescent power consumption of the programming infrastructure" for each of our ASP systems. The second iterations of the Hibernets and Netamorph systems use floating-gate biasing which aid in achieving dense, low-power programmability. High integration of the programming infrastructure enables Netamorph 2.0 to achieve very low quiescent power consumption.

The quiescent current draw of the PCB is just 1.35μA: 276nA for the analog supply, 85nA for the digital supply, and the rest is drawn by the 2.5V regulators and 1.25V reference. Despite the increased size and programmability of Netamorph 2.0, the system-level quiescent current is much lower than our previous Hibernets and Netamorph designs because of the increased attention to the design of the biasing infrastructure. A comparison is shown in Fig. 10.9. In the Figure, "Programmability (bits)" combines the bits of all switches and PLAs, as well as the biasing resolution (in bits) for all analog biases.

For sensors, we have included the Knowles SPW0430 low-power microphone (240μW), the STMicro LIS352 3-axis accelerometer (900μW), and the STMicro LY3200 1-axis gyroscope (12.6mW)—all of which can be completely turned off using on-board switches. Additional sensor inputs can be provided using a 3.5mm stereo audio jack and a 2-pin female header. This combination of sensors makes the PCB useful for prototyping wearable electronics as well as audio/vibration applications.

The PCB includes a header for connecting a TelosB mote. Additionally, we have made a cable for connecting the PCB to Arduino-compatible devices, which softens the learning curve for building applications with the FPAA.

10.4.2 Development Environment

We have developed a two versions of software user interface to aid in the reconfiguration and tuning of the FPAA. The first implementation of the user interface allowed users to wire circuits together in a spreadsheet. Once the user completed the design, the configuration, consisting of the switch settings and the bias values, was converted to a header file by a

Matlab script. This header file was then uploaded to the base-station mote running TinyOS, and then wirelessly transmitted to the remote nodes. The remote node then applied the new configuration to its FPAA. This spreadsheet-based user interface was not practical to scale to the larger Netamorph 2.0 FPAA.

The current implementation of the user interface uses a netlisting language to specify circuits. Subcircuits can be defined and instantiated, thus allowing developers to design hierarchically using a library of circuits. Based upon the netlist, a heuristic-based routing algorithm decides which switches to set in order to connect the circuit. Currently, the user must specify which CAB to use for each primitive device (e.g. when instancing a C^4 bandpass filter, the user must specify which channel to use).

The routing algorithm begins with the CAB with the greatest number of unconnected nets. This CAB will be the most congested, and will therefore be the most susceptible to starting conditions. Within this CAB, the net with the longest connection is routed first so that it does not have to detour around other nets.

This simple development environment has enabled us to be much more productive at creating designs in the FPAA. However, several improvements are envisioned. The path-routing algorithm described above can be improved by including the ability to identify congested areas and choose alternate paths to avoid congestion. The routing algorithm can also be improved by including the ability to separate sensitive hi-impedance nets away from aggressive low-impedance nets. However, coupling between lines in a connection box is primarily through the capacitance across "off" switches, so physical proximity between lines does not factor into the degree of coupling. As a result, sensitive lines are best separated from aggressors by routing them in separate CABs, which is best addressed by automatically placing devices into the CABs as opposed to leaving this responsibility to the user. Beyond implementing automatic placement and improving the routing capabilities, the development environment can be further improved by making it more intention based by, for example, automatically inserting buffers where they will be helpful or using state-machine specifications to synthesize the contents of the lookup tables and the routing in the digital stages.

10.4.3 Compression of FPAA Configuration Files

When designing an FPAA for use in a wireless sensor network, the size of the FPAA is a critical design choice. While it is desirable to have an FPAA that is large enough to create sophisticated ASP designs, care must be taken to minimize the overhead of delivering and storing large configuration files within the network. The naïve approach for handling configuration files is to simply transmit the raw bits that will eventually be shifted into the FPAA. For example, the configuration file for the switch configuration in Fig. 10.10(a) will consist of 64 bits, only three of which are "on," which implies that the configuration is redundant. If this method were scaled to large FPAAs that have 74,000 switches [198], for example, then significant energy would be wasted transmitting and receiving redundant bits.

To address this problem, we have developed a compression method that is inspired by entropy coding, but that is informed by our observations about typical FPAA configurations. Configuration files tend to be small. Therefore, traditional methods which utilize a codebook would have too much overhead (e.g. Huffman coding). Even when considering FPAAs

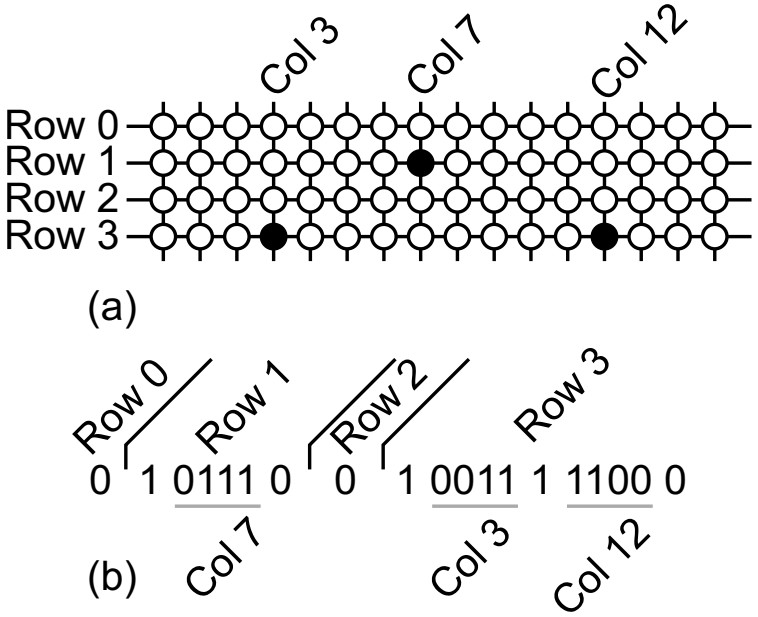

Figure 10.10: (a) Example of a configuration of switches. (b) Implementation of how this configuration would be transmitted using our compression scheme.

of larger scale, ASP algorithms tend to have a parallel nature and are still amenable to compression. An example of this is a large filter bank which utilizes the same operation in each sub-band. This identical operation means that the switch settings are the same in all channels; therefore, it is only necessary to transmit the settings for one channel and then apply those settings to the remaining channels, as we describe below.

Due to redundancies in the switch matrix, most rows tend to have no switch set. Therefore, we begin our row-by-row compression scheme by delineating whether or not any switches are set within a given row. Only if a switch is set in the row do we specify the location of the switch within the row using a four bit identification number. An example compression is shown in Fig. 10.10(b), where the 64-bit configuration of Fig. 10.10(a) is reduced to 19 bits. The size of the compressed configuration depends upon the number of "on" switches N_{on}, and is equal to $5N_{on}+N_{rows}$, where N_{rows} is the number of rows in the FPAA. We have determined experimentally—Fig. 11.7(c)—that the energy for the mote to decode the configuration is 34.1μJ, while the reduction in transmitted data saves 3.5mJ.

We have expanded the compression scheme described above for our larger Netamorph 2.0 FPAA by compressing redundancy in parallel channels and adding compression of floating-gate values. The updates to the compression scheme include:

1. To accommodate the fact that unused terminals will typically be grounded rather than being left to float, a "write mask" is determined. The "write mask" is the most common row word, and all rows are encoded as the XOR with the mask to minimize the number of rows that must be encoded. The mask is the first word in the compressed bitstream.

2. Redundancy across channels is removed. This is accomplished by identifying which

channels in a stage have matching switch settings. These CABs will be encoded once as opposed to encoding each CAB separately.

3. The floating-gate values are compressed. Since biases in a parallel architecture will typically vary linearly or be constant from channel to channel, the biases are taken channel by channel so that the biases can be compressed by only encoding the deltas. If the delta is less than four bits, then only the delta is encoded, otherwise the full value is encoded.

Algorithm 1 shows the decompression process. In this algorithm, the settings can be written into the FPAA as they are extracted, so it is not necessary to hold the entire expanded configuration in the mote's memory. This is important, because the TelosB only has 10KB of RAM.

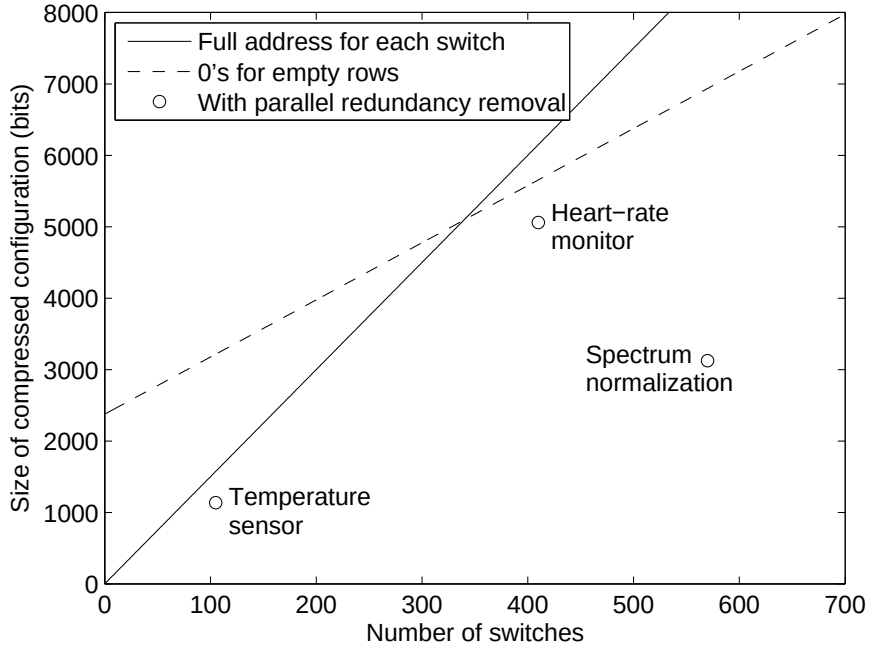

Figure 10.11: Results of applying the compression algorithm to large FPAA designs.

The results of this compression algorithm are shown in Fig. 10.11. The solid and dashed lines show the compression levels which would be obtained either by coding the address of each switch or by using the compression routine in Fig. 10.10, respectively. The circles show bitstream sizes for actual designs (described in the next Section) that were compressed using the routine described above. This new routine has two advantages. First, by only encoding occupied CABs (as opposed to having long runs of '0's for the entire FPAA) we achieve more aggressive compression for small designs, such as the temperature sensor, while minimizing the growth of the configuration size as the number of switches increases, such as for the heart-rate monitor. Second, by removing redundancy from parallelized designs, such as in the spectrum normalization, large designs can be highly compressed.

Algorithm 1 Decompressing the FPAA configuration *Bitstream*

```
Bitmask ← first 12 bits of Bitstream
while "Terminator for Switches" not encountered do
  Channels ← next 8 bits of Bitstream
  while "Terminator for current block of channels" not encountered do
    Stage ← next 4 bits of Bitstream
    Check Stage for "Termination codewords"
    for all Switch rows do
      Word ← 0
      while Next bit of Bitstream = '1' do
        SwitchID ← next 4 bits of Bitstream
        Place '1' at bit SwitchID in Word
      end while
      Word ← Word XOR Bitmask
      Write Word to current row in stage Stage in each channel in Channels
    end for
  end while
end while
while "Terminator for Biases" not encountered do
  Floating-Gate ID ← next 6 bits of Bitstream
  Check Floating-Gate ID for "Termination codeword"
  for all Channels do
    if Next bit of Bitstream = '1' then
      if Next bit of Bitstream = '0' then
        Bias ← PreviousBias + next 5 bits of Bitstream
      else
        Bias ← next 10 bits of Bitstream
      end if
      Write Bias to floating-gate number Floating-Gate ID in the current channel
      PreviousBias ← Bias
    end if
  end for
end while
```

10.5 Applications

10.5.1 Demonstrations of Netamorph 1.0

To illustrate the functionality of FPAA-based ASP designs in wireless sensor networks, we connected our Netamorph 1.0 interface board to a TelosB mote and synthesized several signal-processing circuits on the FPAA. Each of these circuits can be used to generate wake-up signals to turn on the TelosB mote. In each scenario, all reconfiguration commands were sent over the radio through another TelosB mote, which acted as the base station.

10.5.1.1 Rising Frequency Detector

The first system that we demonstrate is the ability of Netamorph 1.0 to perform basic spectral analysis (Figs. 10.12). Here, the FPAA has been configured to analyze a signal's frequency content and detect a rising frequency in the 2-4kHz range. The signal is first filtered through parallel bandpass filters set to center frequencies of 2kHz and 4kHz. The lower-frequency signal (x_0) is then delayed. A cascade of OTAs computes the product of the delayed low-frequency signal with the instantaneous high-frequency signal (x_1), thus providing a measure of simultaneity, reminiscent of the motion-analysis system in [203]. To ensure that static wideband signals do not trigger the detector, the final portion of the circuit pulls the output low when x_0 is high. The resulting output is high only when the x_1 and the delayed version of x_0 are high and the x_0 is not high. As a result, a pulse is generated when the signal is rising in the correct frequency range.

10.5.1.2 Voice-Activity Detector

The next system demonstrates the ability of Netamorph 1.0 to implement a voice-activity detector, based upon the scheme presented in [191] (Fig. 10.13). Audio signals were first passed through the spectral analysis CAB where they were filtered from 10 Hz to 2kHz using a bandpass filter. The envelope of this speech band was then found and passed through another bandpass filter, with corner frequencies at 2Hz and 12Hz corresponding to the phoneme band. The magnitude of the phoneme band was then used to trigger a time-to-voltage converter that would create a ramping voltage when the phoneme band exceeded a specified threshold. The time-to-voltage converter then triggers an event when this ramped voltage exceeded a threshold. While the inclusion of the time-to-voltage converter and the subsequent comparator stage may seem redundant at first, it allowed the device to operate in noisy, non-idealized conditions. The signal at each stage is shown in Fig. 10.13, and it is shown that the speech portion of the input signal was correctly identified in the presence of noise. The overall output can be used to identify to the rest of the sensor node that a signal of interest has been found.

10.5.2 Demonstrations of Netamorph 2.0

The following demonstrations validate Netamorph 2.0 in a variety of sensing applications. All power consumption values (Table 10.3) were obtained by measuring the supply current of the entire FPAA board (at 3V); these values include the power of output buffers,

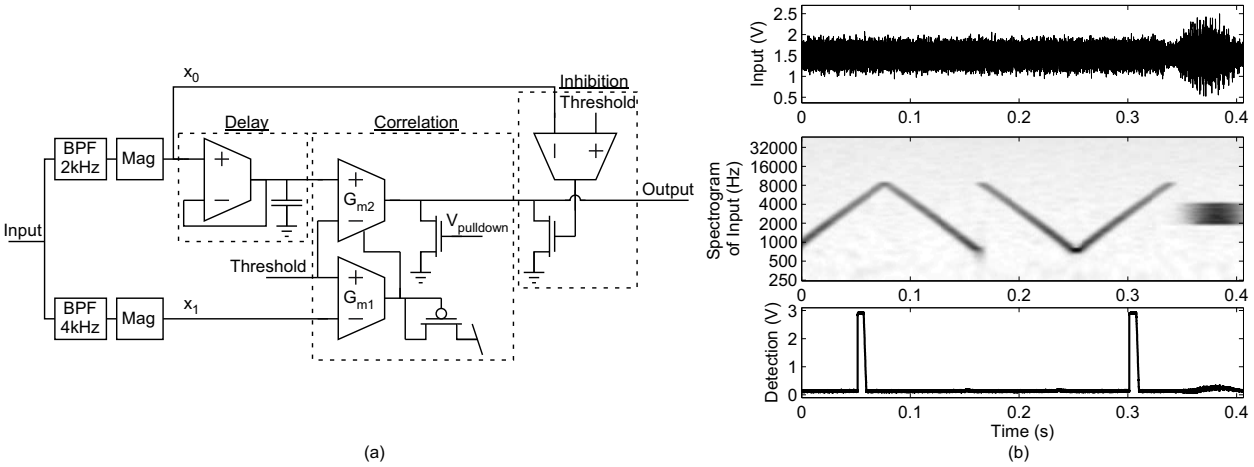

Figure 10.12: (a) Top-level schematic of the rising frequency detector synthesized in Netamorph 1.0. The circuit detects portions of the signal where the frequency content rises in the 2–4kHz range. The "Correlation" stage detects the simultaneous presence of content in the high-frequency band and the delayed low-frequency band. The "Inhibition" stage nulls the output when content is present in the low-frequency band to avoid triggering on wideband signals. G_{m2} is biased by the gate of the attached pFET while the output current of G_{m1} is mirrored through the diode connected FET. Also note that $V_{pulldown}$ is a constant bias used to weakly pull down the output when G_{m2} is shutoff by G_{m1}. (b) Spectral analysis performed by the analog IC. (Top, Middle) The transient plot and the spectrogram plot of the input signal, respectively. The sinusoidal signal, which includes Gaussian noise, varies from 1kHz to 8kHz and concludes with a steady input of ten sine waves ranging from 2kHz to 4kHz. (Bottom) The output stage successfully detects portions of the signal where the frequency content rises in the 2–4kHz range.

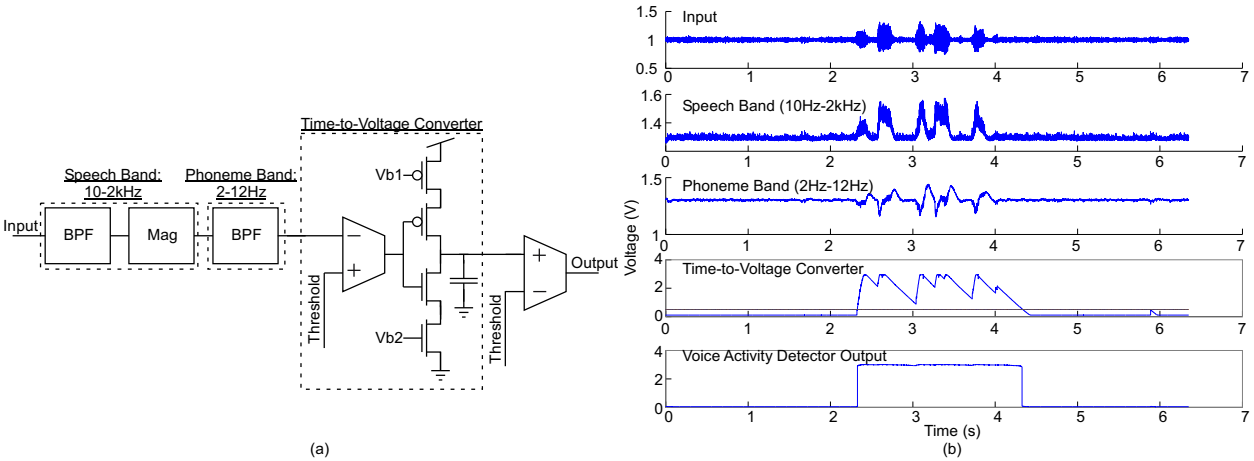

Figure 10.13: (a) Schematic of the voice-activity detection algorithm implemented in Netamorph 1.0. The device triggers an event when the amplitude modulation in the speech band occurs at a rate that is typical of speech. (b)

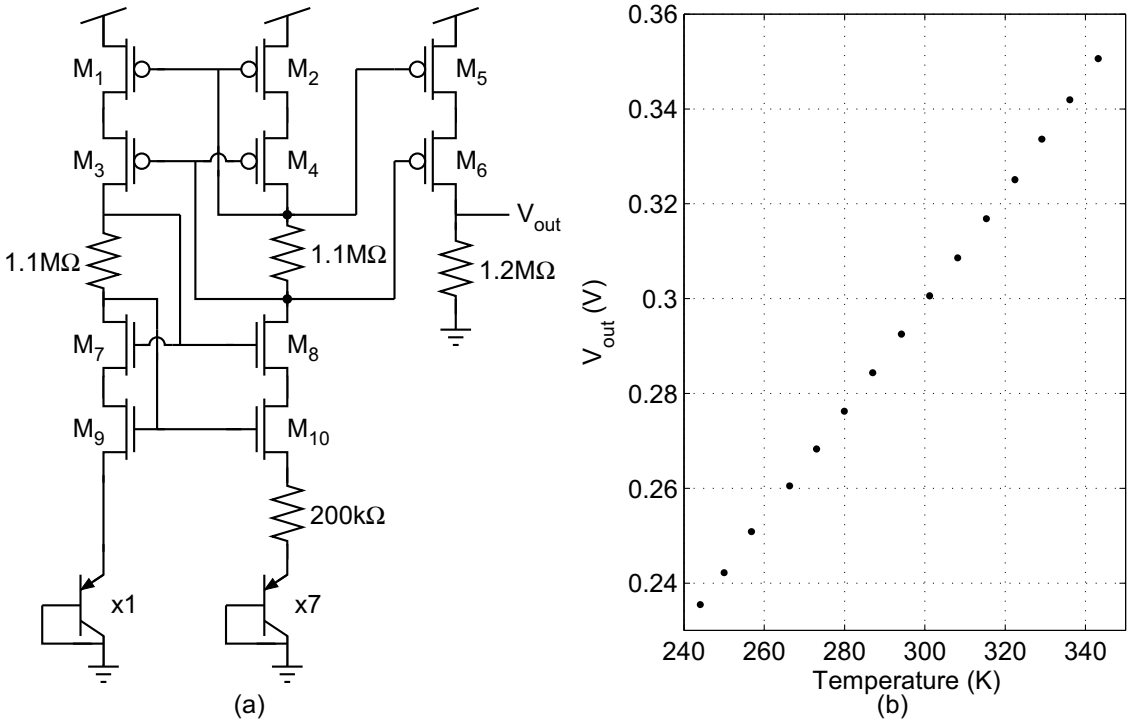

Figure 10.14: (a) Temperature sensor which was synthesized using devices in Netamorph 2.0. (b) Measured output response of the temperature sensor across a 100K temperature range.

regulators, and references, and are thus representative of the power cost to add the FPAA to an embedded system.

Table 10.3: Demonstration results for Netamorph 2.0

Circuit	Run power	Nets	NVMs	Config energy
Temp. sensor	12μW	23	3 (readout buffer)	1.97mJ
Heart-rate	20μW	55	18	5.52mJ
Audio	17.25μW	89	52	10.42mJ

10.5.2.1 Temperature Sensor

Netamorph 2.0 includes a large number of device-level elements to synthesize circuits that are too specialized to include in the computation blocks. The temperature sensor in Fig. 10.14(a) was synthesized as a demonstration. The ratios of the BJTs and resistors were chosen to yield a 1mV/K output. The circuit's measured temperature response is shown in Fig. 10.14(b). Due to second-order temperature dependencies in the BJTs, the output is actually 1.16mV/K. The power consumption is 12μW.

10.5.2.2 Heart-Rate Monitor

Netamorph 2.0 was designed to meet the needs of the first three stages in sensor systems: sensor interfacing, signal analysis, and event detection. These capabilities are demonstrated by synthesizing the heart-rate monitoring system shown in Fig. 10.15(a). The difference amplifier is based on [28] and the system's back-end was inspired by [116]. Each symbol directly maps to a single computational element (with the exception of the shift register, which is a collection of flip flop elements). As a result of the FPAA's mixed granularity, this relatively large system was mapped onto a small number of elements.

The system takes a differential input from ECG probes (in this demonstration the input is generated by a multi-channel DAC). The sensor input is conditioned with two amplification stages and a lowpass filter which removes residual 60Hz noise. The heart rate is extracted by detecting the R wave with a maxima detector, which triggers a time-to-voltage converter, thus yielding the heart period. The system generates an alarm when the heart rate deviates from an acceptable range of 60–110bpm. A windowing operation is used to minimize false alarms by only triggering if multiple recent heart rates are out of range. Measured operation is shown in Fig. 10.15(b). The total power consumption is 20μW.

10.5.2.3 Audio Spectrum Normalization

The architecture of Netamorph 2.0 is amenable to audio and vibration signal processing using an analog filter bank. Information in filter channels is highly correlated and requires subsequent processing to prepare for classification. In Fig. 10.16, we present a circuit implementation of the decorrelation algorithm in [204], which was synthesized in the FPAA. The algorithm's nonlinear inhibition of parallel channels maps efficiently into analog circuitry. The circuit sharpens the filter bank frequency responses and normalizes the channels to create scale-invariant features for classification. The power consumption is 17.25μW. For comparison, [198] describes an FPAA implementation of a single channel of a comparable filter bank algorithm which consumed 34.5μW.

10.6 Conclusion and Future Work

A commonly cited application of FPAAs is sensor interfacing. As the Internet of Things continues to emerge, the quantity of sensing devices will increase while the energy per device will need to decrease. With this in mind, we have developed a large-scale, low-overhead FPAA for systems which have severe power constraints.

To be used in a sensor network, the FPAA's reprogramming infrastructure has been designed to be simple enough to be controlled with 8-/16-bit microcontrollers with minimal impact on microcontroller resources. A tradeoff study of this aspect of the design is given in the following Chapter.

To facilitate the synthesis of large signal-processing systems, we have designed Netamorph 2.0 with a mixture of computational elements and a parallelized signal-flow topology. The system in Fig. 10.16 has 52 analog parameters and 89 nets (a synthesized channel readout scanner is not shown). To our knowledge, this is the largest-scale published system to be implemented in reconfigurable analog fabric.

No consistent figure of merit exists for FPAAs. This is understandable because the intended usage of FPAAs is also not consistent. We will suggest a path toward developing a figure of merit that may allow comparison of a wide variety of FPAAs and may also allow automatic optimization of FPAAs. One recurring figure of merit is bandwidth, which is limited by the switch fabric parasitics. Bandwidth measurements in FPAAs have previously been performed on either synthesized filters or on local lines. Such measurements do not capture the performance of a nontrivial system synthesized in an FPAA. For example, the bandwidth of a local line may be very high but the design of the FPAA may prevent most nets from being routed on local lines; as a result, the typical bandwidth limitations of the switch fabric will be much more severe than the limitations indicated by the specified bandwidth.

A figure of merit that is representative of real designs can be acquired by extracting statistics about parasitics from a corpus of designs, similar to the distribution that we shown in Fig. 11.4 in the next Chapter. Combining such information with modeling of the parasitics of a single switch, it is possible to estimate the typical full-system power cost of maintaining a given bandwidth and signal resolution within the switch matrix. Such a metric is independent of the performance of the CAB elements, but instead combines all of the design tradeoffs that are inherent to FPAA design: parasitics of the switch fabric, number of devices and routing tracks per CAB, organization of CABs by function, and place-and-route algorithms. Further development of this metric can be used to compare FPAAs and to automatically optimize FPAA design parameters.

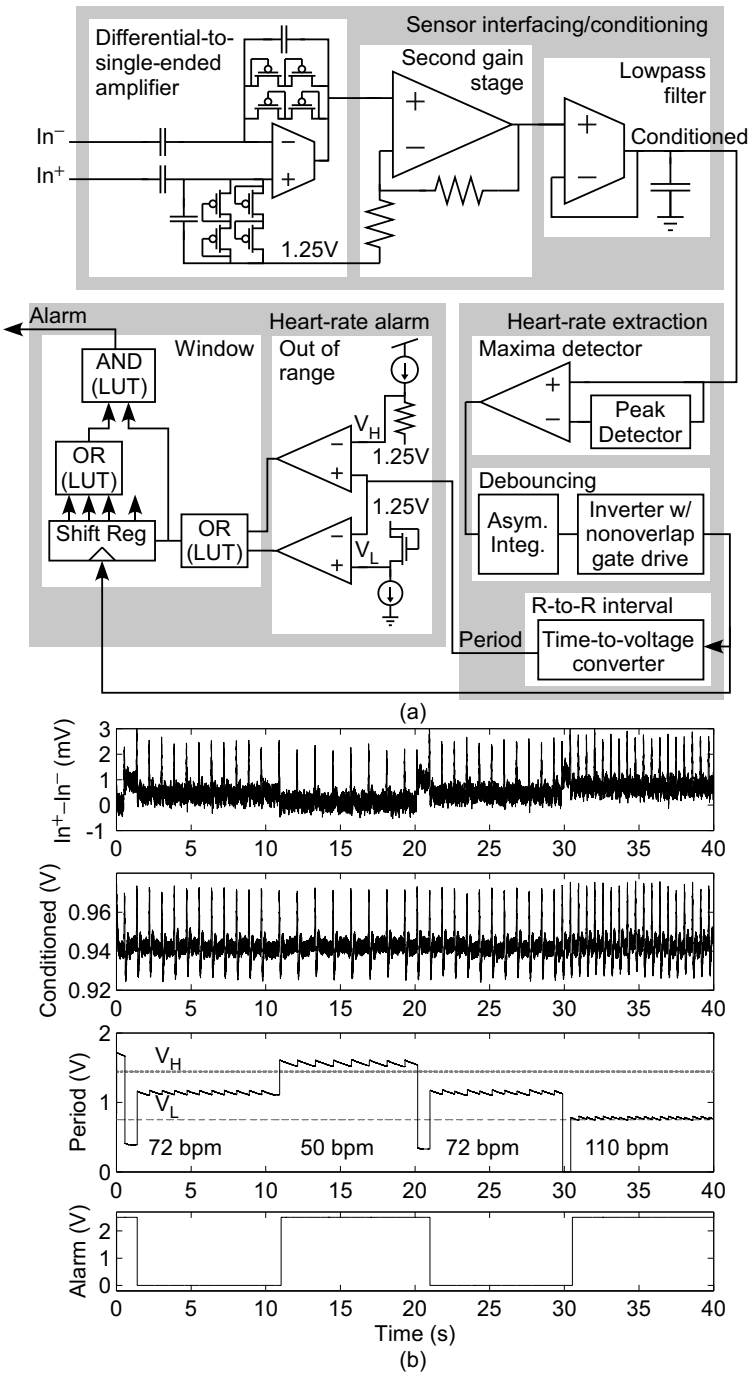

Figure 10.15: (a) Heart-rate monitoring system which was synthesized in Netamorph 2.0. After passing through the conditioning block, a time-to-voltage converter is clocked by the peaks of the R wave to extract the period. The period is compared with user-defined high/low thresholds. If two recent periods are outside of the safe range, then an alarm is generated. (b) Measured response of the heart-rate monitoring system. The input is a 2mV differential cardiac signal with varying heart rate and 200mV 60Hz common-mode noise. The outputs of the conditioning, extraction, and alarm subsystems are plotted. The bottom plot shows successful detection of out-of-range heart rates.

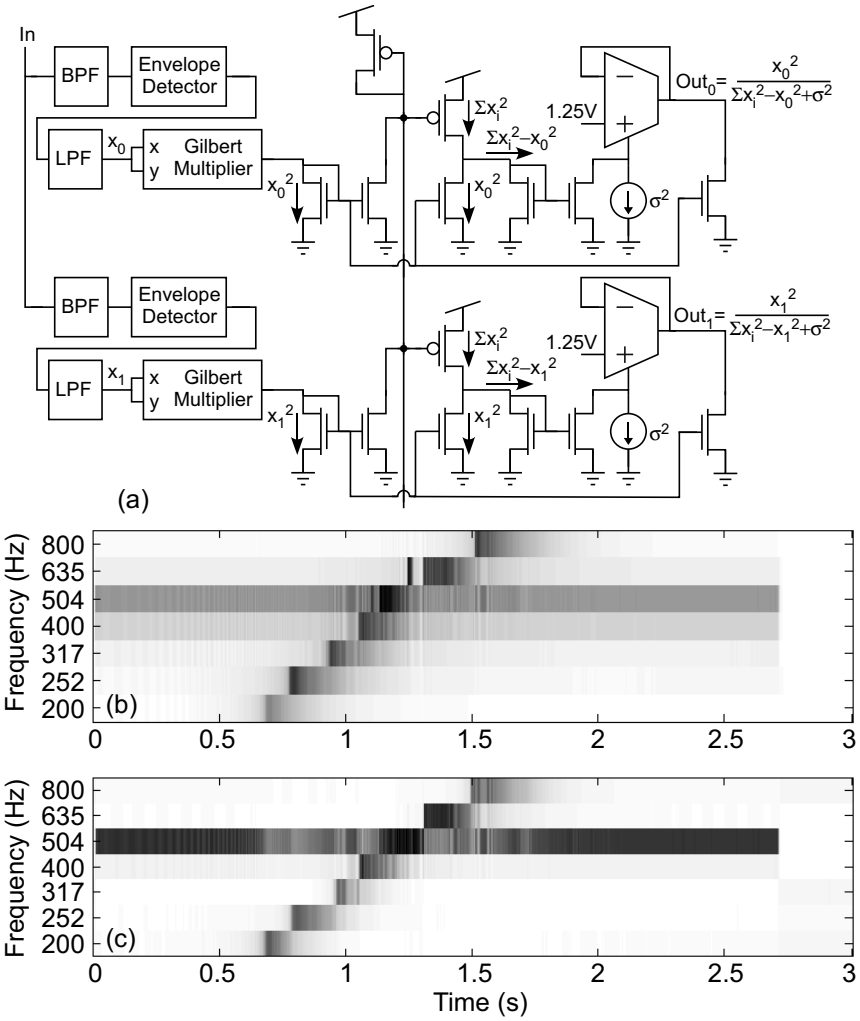

Figure 10.16: (a) Audio spectrum normalization system which was synthesized in Netamorph 2.0. Only 2 of 7 channels are shown for clarity. (b & c) Measured response of the system to a 500Hz tone and chirp combination. (b) Without normalization: x_i. (c) With normalization: Out_i. Note that normalization reduces leakage of the 500Hz tone into neighboring bands, and observe how the 500Hz band is inhibited during the chirp.

Chapter 11

Tradeoffs in Designing Reconfigurable Analog Sensor Interfaces for Wireless Sensing Applications

In the previous Chapter, we described the implementation and application of two "Netamorph" FPAAs (field-programmable analog array) for wireless sensor networks. We showed that these FPAAs extend the range of applications and improve the ease of use of our Hibernets paradigm (Chapter 3), wherein sensor nodes are augmented with analog signal processing.

In this Chapter, we examine the tradeoffs for designing FPAAs for wireless sensors. Section 11.1 describes the background of FPAAs. Section 11.2 analyzes FPAA architecture tradeoffs. Section 11.3 studies the cost of reconfiguring analog circuitry and Section 11.4 examines the implications of the cost of reconfiguration in the higher-level context of wireless sensor networks.

The study of reconfiguration costs in this Chapter was published in the Proceedings of the International Midwest Symposium on Circuits and Systems [205].

11.1 FPAA Trends

D'Mello and Gulak provide an excellent account of FPAA development prior to 1998 [206]. They attribute the emergence of programmable analog circuits to the GAP-01 made by Precision Monolithics in 1982. The GAP-01 (some applications details in [207]), and its sibling peak-detection version the PKD-01 [208], were essentially single-chip computational analog blocks (CABs) containing a comparator, a follower-connected op-amp, a diode (only in the PKD-01), and two transconductance amplifiers that could be switched on or off for field programming. Since then, FPAAs have grown in size and programmability. We have surveyed approximately 30 FPAA designs (including [192,195,196,198,199,201,202,209–226] as well as our two Netamorph designs in this Chapter) for the following illustrations of FPAA trends.

The intended use of FPAAs has changed over time, as illustrated in Fig. 11.1. Two of the earliest FPAAs were designed primarily for synthesizing analog neural networks [201,

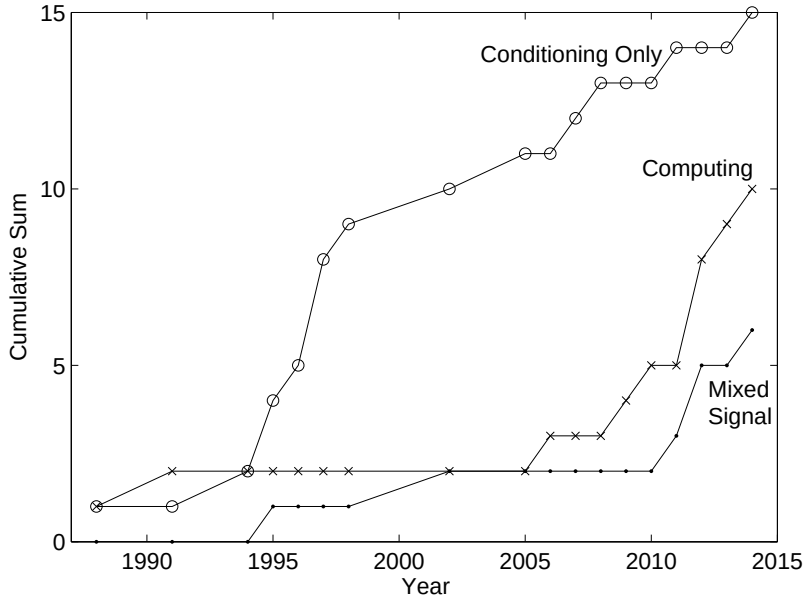

Figure 11.1: The number of FPAAs designed for a given purpose over time. Until recently, most FPAAs were designed only for signal conditioning. But most recent FPAA research has focused on analog signal processing and decision making (i.e. computing) as well as mixed-signal arrays.

209]. Afterwards, approximately ten years passed in which all FPAAs were developed for traditional analog conditioning applications, such as filtering, rectifying, and modulating. In the following discussion and figures, we refer to such FPAAs as "conditioning FPAAs." Two mixed-signal FPAAs were developed during this time and they placed a sharp divide between analog and digital by using two separate analog and digital arrays in which the analog portion was used exclusively for conditioning-type operations [195, 212]. Recent mixed-signal FPAAs have interleaved analog and digital blocks for synthesis of data converters and digitally-assisted analog circuits [224, 225]. We refer to both types as "mixed-signal FPAAs." Recently, work on FPAAs for conditioning is decelerating, while a significant amount of work has focused on reconfigurable analog signal processing ICs that use the dense, low-power computational capabilities of nonlinear analog circuitry to perform highly efficient computing ([72] estimates 10,000-fold improvement over low-power DSPs for certain operations). We refer to such FPAAs as "computing FPAAs," which are not mutually exclusive from mixed-signal FPAAs—in fact, most of the recent computing FPAAs have been mixed-signal: [202, 225] and this work.

All of the surveyed FPAAs were designed using the principle of segmenting circuitry into CABs, although other terminology (e.g. "leafs") may have been used. Figure 11.2 examines the CAB design choices that were made in the surveyed FPAAs. Separate lines are used for "computing" FPAAs and "conditioning" FPAAs because these types clearly follow different trends. For reference, our Netamorph designs are denoted by circles. In general, our group's trend is toward smaller CABs with higher granularity of computational elements.

Although all FPAAs have utilized CABs, a dichotomy exists in the interpretation and

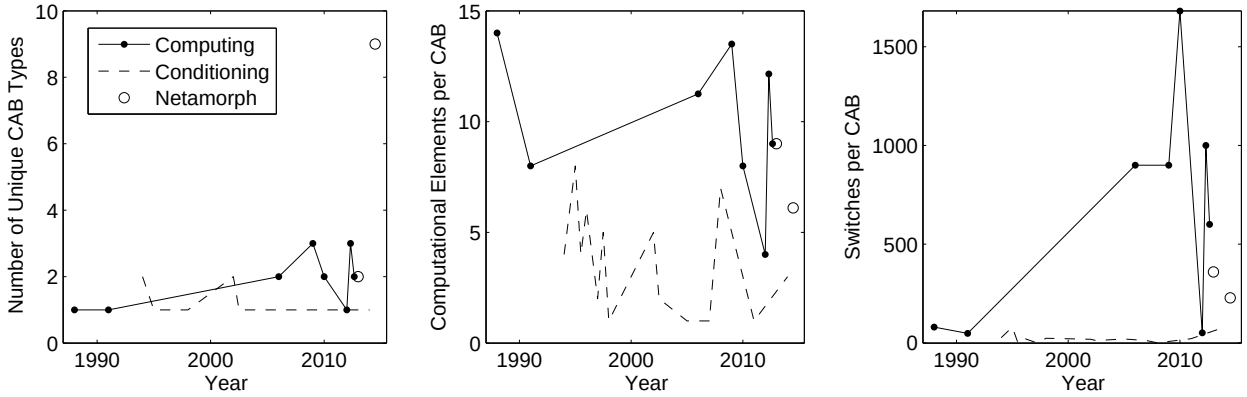

Figure 11.2: Trends in CAB designs. The solid line shows FPAAs designed for computing and the dashed line shows FPAAs designed for conditioning. The circles show our two Netamorph FPAAs. (a) Variety of CAB types. (b) Average number of computational elements in each CAB. (c) Average number of switches used to reconfigure each CAB.

design of CABs.

1. In one interpretation, a CAB is a single "universal cell," such as an adjustable-gain integrator, that can be combined with other like cells in a homogeneous array to synthesize the desired functionality. This approach mimics the mostly homogeneous arrays of latched lookup tables that constitute FPGAs. While this approach simplifies the decomposition of high-level design specifications into circuitry, we argue that the efficient usage of energy and die space that is demanded by low-power, low-cost sensor networks is better served by a more application-customized approach to CAB design.

2. In the other interpretation, a CAB is a collection of different computational elements that can be connected arbitrarily. The elements may be of mixed complexity (or "granularity") to reduce the overall number of switches that are necessary to synthesize more complex designs in the FPAA. Furthermore, some CABs in the FPAA may have a different collection of computational elements to accommodate the expected spatial distribution of processing functions throughout the FPAA.

To facilitate efficient implementation of sensor network functionality, we have adopted the second interpretation of CAB design and have used a mixture of computational granularity ranging from individual transistors and capacitors to filters and timers. By including nine CAB types with a mixture of purposes, as illustrated in Fig. 10.2(a), we are able to accommodate sensor network functionality that ranges from sensor interfacing to information extraction. Figure 11.2(a) compares the number of CAB types that different FPAAs have used. Our use of nine CAB types is a significant shift in FPAA design. Quantifying the value of an increased number of CAB types is a subject of future work, but this design choice has enabled us to synthesize full "sensor interfacing through mixed-signal event detection" systems in one FPAA (e.g. the heart-rate monitor in Section 10.5.2.2), which has not previously been done.

Another issue that arises in FPAA design is the number of computational elements in each CAB. If a small number of elements are used, then a small number of switches are sufficient to connect the elements, which lowers the parasitics of excessive switches. However, with a small number of elements in the CAB, more nets will need to connect to multiple CABs, which will add more switches to the nets and thus negate the benefits. Figure 11.2(b) shows how the number of elements per CAB has varied across FPAAs. We analyze this tradeoff in Section 11.2 and show that most "computing FPAAs" have used too many elements per CAB.

To sufficiently connect the elements in a CAB, the number of switches must increase as the number of elements increases. In the worst case, the number of switches increases quadratically, but in Section 11.2 we show this to be unnecessary. Figure 11.2(c) shows how the number of switches per CAB has varied across FPAAs. The potentially high cost of placing many computational elements in a CAB is clearly evident.

11.2 FPAA Architecture Tradeoffs

As Mead showed in [36], an important consideration for building efficient computing systems is to minimize the connection lengths between computing elements. In general-purpose computing architectures, the memory and processing components are highly separated, thus limiting efficiency because of long connection lengths. Connection lengths can be minimized by adopting a signal-flow architecture with the necessary memory located near the processing elements. Although reconfigurable ICs such as FPGAs and FPAAs are ideal for signal-flow architectures, the high parasitics of the switch fabric significantly increase the cost of wiring. In this Section, we examine how the number of switches per CAB and the number of computational elements per CAB can be chosen to maximize efficiency. Table 11.1 defines the variables that are used in this Section.

Table 11.1: Variables Used in FPAA Analysis

C	Total # of CABs in FPAA
E_{CAB}	Average # of computational elements per CAB
$E_{\text{total}} = CE_{\text{CAB}}$	Total # of computational elements in FPAA
$N_{\text{T,E}}$	Average # of terminals per computational element
$N_{\text{T}} = E_{\text{CAB}}N_{\text{T,E}}$	# of terminals per CAB
N_{R}	# of routing lines for intra- and inter-CAB connections (per CAB)
R	Rent exponent
$S_{\text{CAB}} \geq N_{\text{T}}N_{\text{R}}$	# of switches per CAB
$S_{\text{total}} = CS_{\text{CAB}}$	Total # of switches in FPAA
S_{min}	Minimum # of parasitic switches connected to a net
S_{avg}	Average # of parasitic switches connected to a net

11.2.1 Applying Rent's Rule to FPAA Design

One crucial issue in FPAA design is balancing the number of CAB terminals, connection-box switches, and routing lines. Too few switches and routing lines will limit the ability to connect many nets in a small region of the FPAA. Designs will therefore span extra CABs because one CAB cannot accomodate enough nets, this results in underutilization of computing elements within the CABs and excessive parasitics from unnecessarily long nets. On the other hand, too many switches wastes space and adds unnecessary parasitics to short nets. In both cases, improper balancing of the number of CAB terminals, connection-box switches, and routing lines results in excessive parasitics and underutilization of chip space.

This issue can be better understood using Rent's rule [227] (previously applied to FPAA design in [209]), which describes the relationship between the number of communication terminals of a subblock (i.e. the number of routing lines for intra- and inter-CAB connections, N_{R}) and the number of computing elements within that subblock (i.e. the number of computational elements in each CAB, E_{CAB}). Rent's rule is an empirical observation that the number of communication terminals scales with the number of computing elements as follows

$$N_{\mathrm{R}} = N_{\mathrm{T,E}} E_{\mathrm{CAB}}^{R} \tag{11.1}$$

where $N_{\mathrm{T,E}}$ is the average number of terminals per element and R is the Rent exponent. If $R = 1$, then the number of routing lines equals the number of CAB terminals, which implies that none of the gates within the block connect to each other. In an FPAA connection box, we would then require a routing line for each terminal of each computational element. For each terminal to connect to a unique routing line, this case requires a switch matrix with $N_{\mathrm{R}} N_{\mathrm{T}} = N_{\mathrm{T,E}}^2 E_{\mathrm{CAB}}^2$ switches—the number of switches grows with the square of the number of terminals, which is known as a "full-crossbar" switch matrix. If the design contains any placement optimization, then we expect connections beween neighboring elements to be most common, so the number of routing lines is less than the number of CAB terminals and $R < 1$. The value of R depends on the computing architecture and the wiring capabilities of the process, with typical empirical values for ICs ranging from 0.5 to 0.75 [209].

In Table 11.2 we compare the Rent exponents of our FPAA designs. We found that Netamorph 1.0 had excessive routing, which is illustrated by $R = 0.82$ being outside of the typical range of 0.5 to 0.75. In Netamorph 2.0, the routing was further optimized with R in the typical range, and the CAB-level routing has generally been found to be neither excessive nor insufficient. However, the numbers in the Table are averages and we have found the digital-stage CAB to be an outlier. With three computational elements and an average of 5.33 terminals per element, the digital-stage CABs have a Rent exponent of $R = 0.48$, which is consistent with the fact that the digital stages are the worst choke point for routing. This illustrates an important issue when designing FPAAs with a mixture of CAB types: not only should the number of CAB terminals be constant throughout the FPAA, but the Rent exponent should also be constant throughout the FPAA.

In summary, it is recommended to use (11.1) to determine the number of routing lines. The routing lines are the lines that are not fixed to nets—i.e. not ground, V_{dd}, or midrail.

Table 11.2: Rent Exponents of Netamorph FPAAs

	Elements/CAB E_{CAB}	Terminals/Element $N_{\mathrm{T,E}}$	Routing/CAB N_{R}	Rent exponent R
Netamorph 1.0	9	2.39	14	0.82
Netamorph 2.0	6.1	2.62	9	0.68

11.2.2 Designing CAB Size

A crucial parameter in FPAA design is the number of computational elements in each CAB. If a small number of elements are used, then a small number of switches are sufficient to connect the elements, which reduces the parasitics from excessive switches. However, with a small number of elements in the CAB, more nets will need to connect to multiple CABs, which will add more switches to the nets and thus negate the benefits of small CABs. We will examine how the average number of parasitic switches per net depends upon the CAB size.

First, consider the effect that the number of elements per CAB, E_{CAB}, has on the total number of switches in the FPAA, S_{total}. In our FPAA, approximately 60% of the area is devoted to switches, and other recent FPAAs have devoted greater portions of their area to switches, so variations in switch count contribute significantly to the overall FPAA size. In this thought experiment, we will maintain a constant value for the total number of computational elements in the FPAA, E_{total}, so the variation in the overall size of the FPAA is only due to a variation in the number of switches.

Since the total number of computational elements in the FPAA is constant, the number of CABs in the FPAA is $C = E_{\mathrm{total}}/E_{\mathrm{CAB}}$. The total number of switches in the FPAA is then $S_{\mathrm{total}} = S_{\mathrm{CAB}}E_{\mathrm{total}}/E_{\mathrm{CAB}}$. The number of switches in a CAB is the sum of three types of switch:

1. The switches in the connection box that are used to route between computational elements within one CAB—illustrated in Fig. 10.3(b). This is the number of routing lines multiplied by the total number of terminals per CAB. Using (11.1) yields $N_{\mathrm{R}}N_{\mathrm{T}} = N_{\mathrm{T,E}}^2 E_{\mathrm{CAB}}^{1+R}$.

2. The switches in the connection box that are used to connect the computational elements to ground, V_{dd}, and midrail. This is three switches per terminal: $3N_{\mathrm{T,E}}E_{\mathrm{CAB}}$.

3. The switches in the switch box of an "island-style" archiecture, which typically have six switches—illustrated in the 4-point switch in Fig. 10.2(c). One 4-point switch is needed for each routing line, so the number of switch box switches is $6N_{\mathrm{T,E}}E_{\mathrm{CAB}}^R$.

Combining these provides the number of switches in a CAB:

$$S_{\mathrm{CAB}} = N_{\mathrm{T,E}}^2 E_{\mathrm{CAB}}^{1+R} + 3N_{\mathrm{T,E}}E_{\mathrm{CAB}} + 6N_{\mathrm{T,E}}E_{\mathrm{CAB}}^R \tag{11.2}$$

The total number of switches in the FPAA is then

$$S_{\mathrm{total}} = S_{\mathrm{CAB}}\frac{E_{\mathrm{total}}}{E_{\mathrm{CAB}}} = E_{\mathrm{total}}N_{\mathrm{T,E}}\left(N_{\mathrm{T,E}}E_{\mathrm{CAB}}^R + 3 + 6E_{\mathrm{CAB}}^{R-1}\right) \tag{11.3}$$

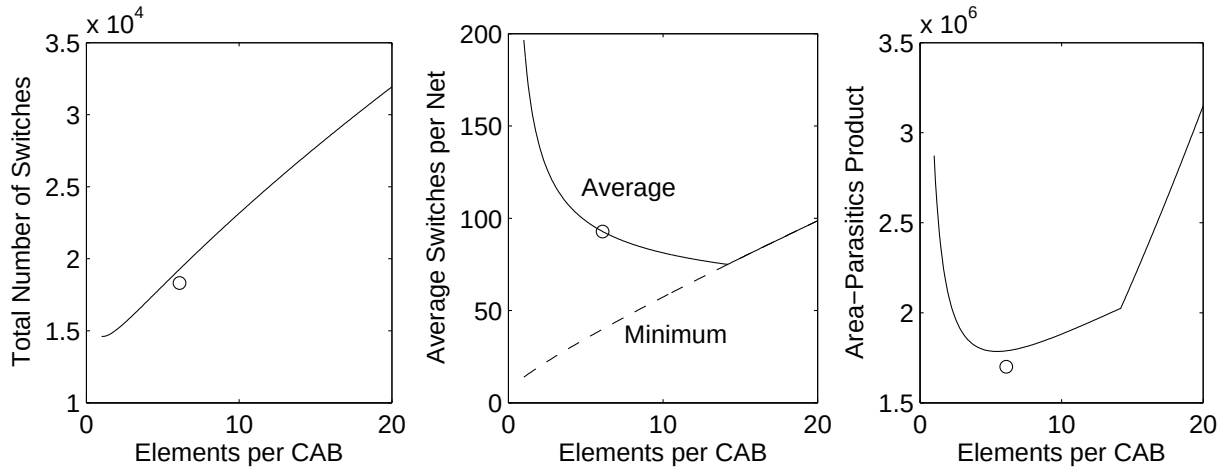

Figure 11.3: Effect of CAB size on FPAA size and performance. The circles show Netamorph 2.0. (a) Increasing the number of computational elements per CAB increases the total number of switches in the FPAA, even though the total number of computational elements in the FPAA is constant. (b)"Average" and "Minimum" number of parasitic switches on a net as a function of the CAB size. (c) The best tradeoff between the "Area" (total number of switches) and the "Parasitics" (average number of switches per net) occurs at approximately five computational elements per CAB.

Figure 11.3(a) shows how the total number of switches depends upon CAB size. We used $E_{\text{total}} = 480$, $N_{\text{T,E}} = 2.62$, and $R = 0.68$, which are the values in our FPAA. The size of the FPAA grows significantly as the CAB size increases, despite the fact that the total number of computational elements is constant. This illustrates one disadvantage of large CAB sizes.

Now, consider how the CAB size affects the number of parasitic switches. Referring to Fig. 10.3(b), a basic connection between two terminals in a CAB will have

$$S_{\min} = N_{\text{T}} + N_{\text{R}} + 12 \tag{11.4}$$

parasitic switches. The constant "12" incorporates 3 switches on each horizontal line for ground, V_{dd}, and midrail, as well as 3 switches on the top and bottom of the vertical line for the 4-point switches in the switch boxes. $S_{\min}$ is the minimum number of parasitic switches for a connection. For FPAAs with small CABs, many nets will make connections in multiple CABs and thus incur parasitics of multiples of this minimum. The number of CABs that various nets connect to depends on the design that is synthesized in the FPAA, but by examining an ensemble of designs we can extract representative statistics. We have extracted empirical statistics on the number of CABs each net connects to from the three demonstrations in Section 10.5.2 and also from an accelerometer double-click detection design. A total of 183 nets are represented. Figure 11.4 shows a histogram of the number of nets that connect to different numbers of CABs. We have found that this empirical data matches a geometric distribution with probability that a net connects to c CABs of

$$P_c = p(1-p)^{c-1} \tag{11.5}$$

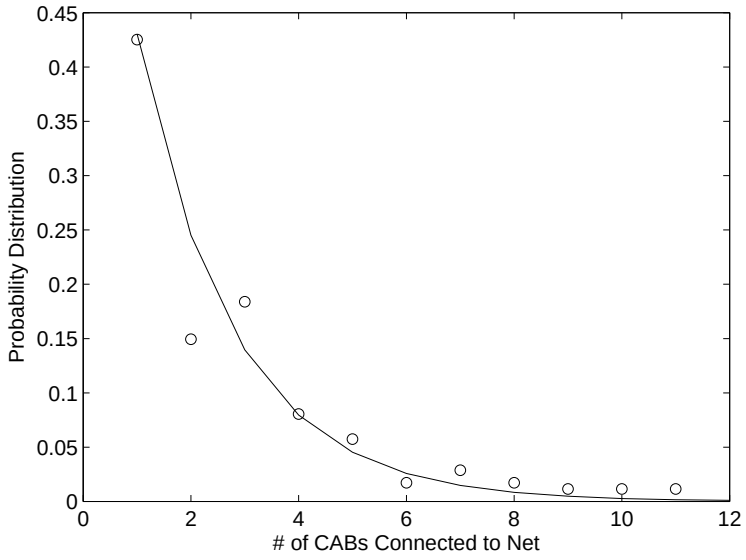

Figure 11.4: Number of CABs per net.

and $p = 0.43$. An interpretation of this distribution is that for a given number of CABs, the net has probability p of finishing its connections and probability $1 - p$ of needing additional CABs. The expected number of CABs that a net will connect to is $1/p = 2.33$. So the average number of parasitic switches on a net is $S_{\min}/p \approx 107$.

p is the probability that a net will only connect elements in a single CAB. If the CABs have six computational elements and one computational element is removed, then it becomes 5/6-times less likely that a net will only connect to computational elements in a single CAB, so p is scaled by the amount that the CAB size is scaled. Our empirical data shows that $p = 0.43$ when $E_{\text{CAB},0} = 6.1$. Combining this information with the above reasoning, we can obtain how the average net parasitics varies with E_{CAB}.

$$S_{\text{avg}} = S_{\min} \frac{E_{\text{CAB},0}}{pE_{\text{CAB}}} \tag{11.6}$$

which becomes

$$S_{\text{avg}} = \frac{E_{\text{CAB},0}}{pE_{\text{CAB}}} \left(N_{\text{T,E}} E_{\text{CAB}} + 2N_{\text{T,E}} E_{\text{CAB}}^{R} + 12 \right) \tag{11.7}$$

Figure 11.3(b) shows how S_{avg} and $S_{\min}$ vary with CAB size. If we co-consider the area of the FPAA (i.e. total number of switches) and the average parasitics (i.e. average number of parasitic switches on each net), then we obtain the "Area-Parasitic Product" shown in Fig. 11.3(c). Minimizing the "Area-Parasitic Product" provides the best tradeoff between FPAA size and performance. The minimal value occurs when the number of computational elements per CAB is approximately five. This result is supported by [228], which simulated benchmark tests on FPGAs with different sized logic blocks and found that a cluster of four to six LUTs minimized the area-delay product.

Future work on this analysis should take into account the heterogeneous structure of a parallelized FPAA architecture, wherein CAB types are the same in each channel but different in each stage, and connections between different channels are rarer than connections

amongst different stages in one channel. Also, future work should develop a corpus of benchmark tests from which to extract statistics about connectivity within FPAAs.

11.3 The Cost of Analog Reconfiguration

The optimal design and usage of FPAAs for wireless sensing depends upon the cost of reconfiguration, which raises tradeoffs such as the universality and power consumption of the FPAA, as well as the volatility of the configuration. General FPAA design choices were previously outlined in [206]. This Section studies the circuit-level and node-level costs of FPAA reconfiguration and what these costs imply for using FPAAs within wireless sensing, where energy consumption is of paramount importance. To aid this study, we have fabricated two 1280-switch FPAAs in 0.35μm CMOS—one with volatile switches and one with nonvolatile switches—and measured the energy while reconfiguring with an off-the-shelf sensor platform. Throughout this Section, we assume a 3V supply voltage as is typical in battery-operated sensor nodes.

In an FPAA, reconfiguration is achieved via programmable connections in the connection box and the switch box (Fig. 10.1), which are used for local routing and global routing, respectively. Programmable connections can be created using unbuffered conductive switches (such as pass transistors) or buffered switches (such as current mirrors or voltage followers). We will focus on unbuffered switches since they are more versatile and have no run-mode power consumption. Two previously used unbuffered switches that achieve rail-to-rail operation are a transmission gate (T-gate) controlled by an SRAM cell [Fig. 11.5(a)] and a floating-gate (FG) pass transistor controlled by the nonvolatile charge that is stored on its gate [Fig. 11.5(b)] [198]. The FG transistor's gate is not constrained to be within the supply rails and so can be programmed with enough overdrive to achieve a low resistance across the range of operation [229]. In both FPAAs, an SPI block selects the column to write, and then data bits are either latched into the SRAM memory cells or are used to select the FGs to program.

11.3.1 Equivalent Switch Resistance

The simulated resistance of T-gate and FG switches is shown in Fig. 11.5(d). For both switch types, the resistance varies with the common-mode voltage due to the body effect. In the T-gate, the nFET is minimum size and the pFET is sized for symmetric drive strength. The pFET in the FG is the same size as in the T-gate. Note that the kΩ-range switch resistance does not create a significant voltage drop for the nA-range currents that are common in ultra-low-power analog computation systems. Wider switches may be used for resistance-sensitive circuitry; matching resistance between T-gates and FGs is not impacted by changes in dimension when their relative size remains constant.

We define the switches to have equivalent resistance when they have the same average resistance across the supply rails. Under the sizing conditions specified above, the T-gate has an average resistance of 1.8kΩ. The ratio of the FG's mean resistance to the T-gate's mean resistance is shown in Fig. 11.5(e); $V_{fg} = -1.48$V is required to match the T-gate's average resistance. Therefore, in this paper, we specify the FG voltage for an "on" switch to

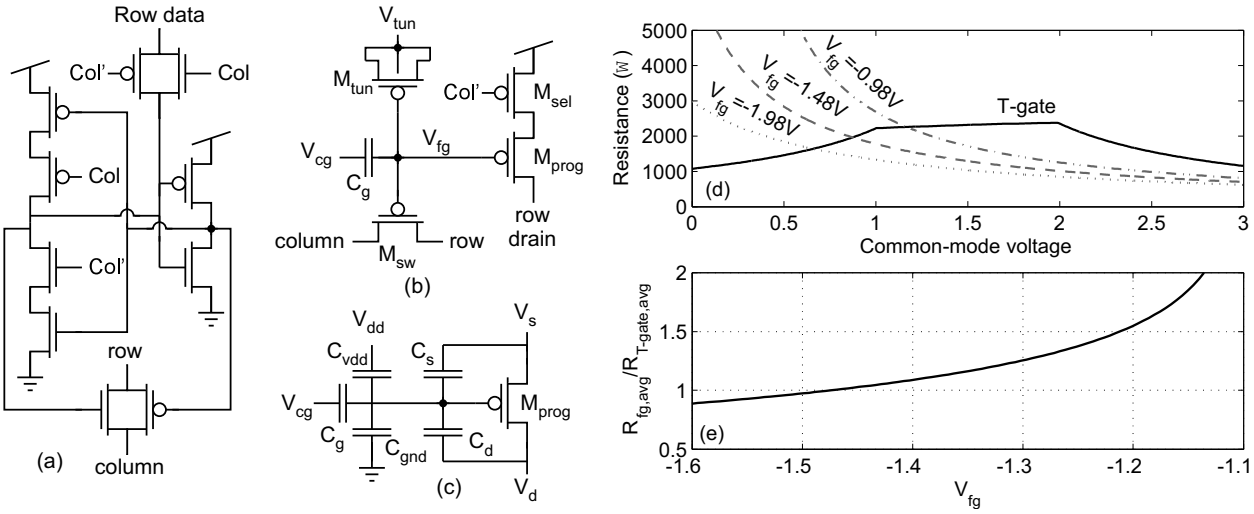

Figure 11.5: (a) Example SRAM-controlled transmission-gate switch. *Col* (*Col'*) selects a column of switches to be rewritten. Switch on/off settings are loaded in parallel through *Row data*. *Row* and *column* are the analog routing paths which may be connected through the switch. (b) Example floating-gate transistor switch. C_g=45fF. M_{tun}=0.4μm x 0.4μm. (c) Capacitive coupling of terminals onto the floating gate. (d) Simulated comparison of the resistance of T-gate switches and FG switches. (e) Simulated comparison of the mean resistance of a T-gate switch to the mean resistance of an FG switch.

be $V_{fg,on} = -1.5$V and the voltage for an "off" switch to be $V_{fg,off} = 3$V. However, despite having equal mean resistance, the FG's sharply increasing resistance near ground may be unacceptable for some applications.

11.3.2 Erasing a Floating-Gate Switch Matrix

Since the majority of switches in an FPAA switch configuration will be "off," an efficient way to program an FG matrix is to globally erase all switches (by removing electrons from the FGs), and then write only the switches that must be turned on. In the cell in Fig. 11.5(b), elecrons are removed by raising V_{tun} to a sufficient voltage to cause electrons to tunnel through the thin oxide of M_{tun}. This mechanism is described by the Fowler-Nordheim tunneling equation

$$I_{tun} = \alpha W L e^{\frac{-\beta t_{ox}}{V_{tun} - V_{fg}}} \tag{11.8}$$

where I_{tun} is the tunneling current through M_{tun}, $\alpha = 185.5\frac{A}{\mu m^2}$ and $\beta = 32.8\frac{V}{nm}$ are parametric fits, t_{ox} (7.7nm for 0.35μm) is the oxide thickness, and W and L are the width and length of the tunneling junction. Using (11.8), we determined that $V_{tun} = 12.5$V is sufficient to tunnel all switches to the off state ($V_{fg,off} \approx 3$V) within 100μs. This high-voltage pulse was generated using an on-chip regulated charge pump; measurements of the voltage pulse and supply current are shown in Figs. 11.6(a)&(b), respectively. The total energy to erase the entire switch matrix is 183nJ. Since the power of the high-voltage generator significantly exceeds the power delivered to the tunneling junction, minimizing the duration of the pulse

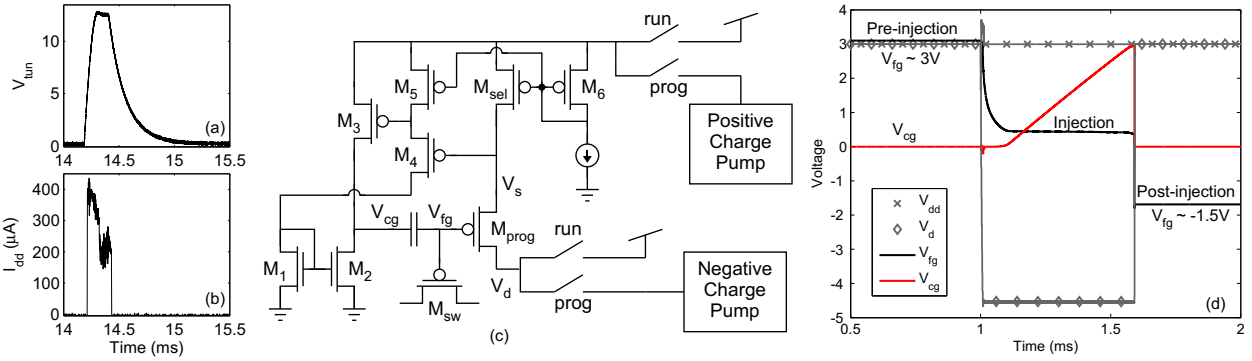

Figure 11.6: (a) Measured high-voltage pulse from the on-chip tunneling charge pump. (b) Measured supply current during the high-voltage pulse. The total erase energy was 183nJ. (c) Illustrative injection circuit for "turning on" FG switches. Transistors M_1–M_5 implement negative feedback from V_s to V_{cg}, thus holding V_s and V_{fg} at the desired voltages during injection. (d) Simulation of the illustrative injection circuit. The simulation was performed with device-level implementations of the charge pumps. The total program energy, including the charge pumps' ring oscillators and regulation circuitry, was 152nJ.

is crucial to minimizing the energy consumption.

11.3.3 Writing a Floating-Gate Switch

After tunneling has been used to remove electrons from all FGs, hot-electron injection is typically used to place electrons onto, and thus "turn on," selected FGs. Injection is commonly used to selectively program standard CMOS FGs because, unlike tunneling, the programming voltages are low enough that standard devices can be used to isolate FGs. Regardless, programming the switches to $V_{fg,on}$ presents a challenge since the FG must be programmed very far below ground to achieve low "on" resistance.

Injection can be modeled using [160]

$$I_{inj} = \alpha I_s (V_{gd} + V_T) e^{\frac{-\beta}{V_{gd}+V_T}} \tag{11.9}$$

where $\alpha = 9$ and $\beta = 80$ are parametric fits for 0.35μm, and V_T is the threshold voltage. A large V_{gd} ($\geq$4.5V) is necessary to achieve fast and efficient injection. To accommodate this high V_{gd} voltage, the supply voltage is typically raised during injection. With the drain connected to ground, the FG will need to be $\geq$4.5V to program quickly. After injecting, the FG must be shifted down 6V to reach $V_{fg,on} = -1.5$V so that the switch has low "on" resistance. This FG voltage shift corresponds to a V_{cg}-referenced shift of $\frac{C_T}{C_g}$6V, which is approximately 8.5V for typical capacitance values. This number illustrates that raising the supply voltage for injection is an inefficient method to program negative FG values—to maintain safe supply voltages, we will be limited to low V_{gd} at the end of the injection cycle, and thus slow programming. For our estimate of the energy to write a switch, we will assume the use of negative drain voltages.

Using the FG switch capacitive-coupling model shown in Fig. 11.5(c), we can determine the FG terminal voltages during injection that will correspond to an "on" switch. In run

mode, $V_{cg} = 0$V, $V_s = V_d = V_{dd}$, and $V_{fg} = -1.5$V. We can solve for the necessary program-mode FG voltage,

$$V_{fg,p} = \frac{V_{fg,on} + V_{dd}(\frac{C_g}{C_T} - \frac{C_s}{C_T} - \frac{C_d}{C_T}) + \frac{C_s}{C_T}V_{sg,p} - \frac{C_d}{C_T}V_{gd,p}}{1 - \frac{C_s}{C_T} - \frac{C_d}{C_T}} \tag{11.10}$$

were $V_{sg,p}$ and $V_{gd,p}$ are the program-mode source-to-gate and gate-to-drain voltages. If we use $V_{gd} = 5$V for fast and efficient programming and want to program the FG within 1ms, then (11.9) gives $I_s \approx 7\mu$A (which yields $V_{sg,p} \approx 1$V). Using (11.10) with $\frac{C_g}{C_T}$=0.7 and $\frac{C_d}{C_T}$=$\frac{C_s}{C_T}$=0.02, we obtain the following program-mode approximate terminal voltages: $V_{fg,p} = 0.41$V, $V_{d,p} = -4.59$V, and $V_{s,p} = 1.41$V.

Figure 11.6(c) shows a complete demonstrative circuit for writing switches using this method. The circuit was designed to hold the FG terminals at the above-mentioned values during injection. A regulated negative charge pump generates the drain voltage. Since the FG is initially "off" at 3V, a positive charge pump is used to generate a short pulse (10μs) on the supply line to start up injection. A full transistor-level simulation for this switch-writing scheme is shown in Fig. 11.6(d). The total energy to turn on a single switch, including the voltage generation circuits, is 152nJ.

11.3.4 Energy Costs of Volatile and Nonvolatile Switches

To determine the reconfiguration cost of an SRAM-based FPAA, we measured the supply current of an FPAA while it was being reconfigured by a sensor platform. Figure 11.7(a) shows this measurement. The total energy is 80.4nJ, or $\approx$ 1nJ/write. This energy is primarily from the SPI block that decodes the incoming serial stream.[1] The FG FPAA will incur the same cost to interpret the serial stream.

In the previous Section, we determined the cost of erasing an FG switch matrix and the cost of writing a single switch. The total cost to reconfigure an FG switch matrix depends upon the number of "on" switches. The percentage of switches that are "on" depends upon the complexity of the circuit being synthesized and the design of the FPAA. In our 1280-switch FPAA, we have found that few configurations use more than 5% of the switches. Additionally, although the overhead of generating high injection voltages can generally be amortized through parallel programming, the sparse distribution of "on" switches within a switch matrix confounds the energy reduction of parallel programming. The energy costs for the switches are summarized in Table 11.3. In Section 11.4, we will interpret these results in the context of wireless sensing.

11.3.5 Other Considerations Regarding Switches

Density: Although the SRAM cell has more devices in the cell, the FG cell requires a dedicated n-well for M_{tun} (which consumes space), and also requires that C_g is large enough

[1] An approximate calculation of the SPI energy shows that this measurement is reasonable: The SPI shift register has N_{stages}=23 stages. Each stage has four logic gates, which we will estimate to have 50% probability of switching on each clock cycle, so the number of switching gates is $N_{switching}$=2. The capacitance of a logic gate is $C \approx$5fF. The energy estimate for an SPI transfer is thus $E = N_{stages}N_{switching}CV_{dd}^2 \approx$2nJ/write.

to dominate the capacitance on the floating node. Consequently, our layouts for the cells were the same size (20.4μm x 8.8μm).

Scaling: Charge leakage is a concern for FGs in deeply scaled standard CMOS, particularly when the FG voltages exceed the supply rails. Thick-oxide devices may retain charge longer, but lose the benefits of scaling. Low-leakage SRAM circuits will be needed for deeply scaled SRAM FPAAs; however, the small number of switches, low density, and low write speeds that are acceptable for an FPAA (compared to a memory array) make leakage less of a challenge.

Reliability: Much more stress is placed upon FGs in switch matrices than in Flash memory or analog circuit trimming. Much more charge is passed through the oxide on each programming iteration and the "on" switches have a high electric voltage across the oxide in run mode (4.5V).

Computation: The tunable conductance of FG switches allows them to be used as computational elements, thus improving the die utilization of FPAAs [230].

Capacitance: Since the T-gate's nFET is much smaller than the pFET, an equivalent pFET FG switch does not have significantly less capacitance than an equivalent T-gate. However, an equivalent nFET-based FG switch would achieve significantly less capacitance. The problem with nFET-based FG switches is that tunneling/injection turns them on/off. So we have to program all of the off switches, which is a larger number than the on switches, and so has high energy cost.

Summary: SRAM FPAAs have clear advantages in terms of reliability, CMOS scaling, and reconfiguration energy (≈123x less than FG FPAAs). However, we will show in the next Section that the switch reconfiguration cost is a small part of the system's overall reconfiguration cost, meaning that FG switches are viable when nonvolatility and/or switch computation are beneficial.

11.4 System-Level Implications of Reconfiguration

To place the FPAA's configuration energy into the system context, we measured the energy of a standard low-power wireless sensing platform (Telos mote [13]) as it received a 1280-bit FPAA configuration over the radio and then programmed the configuration into an SRAM-based FPAA. The results are shown in Fig. 11.7(b). The energy for the mote to wirelessly receive the configuration was 5.3mJ and the energy for the mote to transfer the configuration serially into the FPAA was 0.331mJ (E_{ser} = 4.1μJ/column). This serial transfer energy (≈1.4μJ/byte) is high because the microcontroller's SPI modules are engaged by the mote's radio and external memory, so we had to implement SPI in software using the general I/O pins. In contrast, [231] reports 84nJ/byte for an optimized serial transfer using the same microcontroller. The node-level reconfiguration energy is dominated by the receive energy and by the serial transfer energy. The FG FPAA has lower serial costs in Table 11.3 since only the "on" switches (typically 5%) require a serial transfer after the global erasure. Similarly, a global reset would reduce the serial cost for an SRAM FPAA.

Since the primary energy cost is wireless reception of the configuration, we developed an entropy-coding algorithm to compress the configuration file (details in Section 10.4.3). For typical FPAA configurations, we achieve a compression factor of >4. Since many large-scale

Table 11.3: Summary of Reconfiguration Costs

# Switches	Typ. Sw. Usage	Mote Serial Trans.
1280	5%	E_{ser}=4.1μJ/column
FG Erase	FG Write	SRAM Write
$E_{fg,tun}$=183nJ	$E_{fg,w}$=152nJ/sw	$E_{sram,w}$=1nJ/column
FPAA Type	Total Reconfiguration Energy	
	FPAA-only	FPAA w/ Mote Serial
FG	9.9μJ	276μJ (only prog. "on" columns)
SRAM	80.4nJ	331μJ (prog. all columns)

analog signal-processing systems have identical parallel channels, higher levels of compression are achievable in larger systems. Figure 11.7(c) shows the measured supply current while receiving the compressed configuration (1.8mJ) and then decompressing the configuration in the mote (34.1μJ). The energy was reduced by approximately 65%

Many wireless sensors are powered by unreliable energy sources, thus volatile FPAAs will incur a cost for restoring configurations. In contrast to SRAM-based switches, FG switches are nonvolatile and therefore do not need to be reprogrammed after a power loss. The lower cost of reprogramming SRAM compared to FGs implies that volatile switches are preferable when the frequency of power outages, f_p, is rare compared to the frequency of fresh reconfigurations, f_r. Assuming that the energy to write SRAM ($E_{sram,w}$) is much smaller than the energy to write a floating-gate ($E_{fg,w}$) and assuming block erasure for both types, SRAM will be lower cost when $\frac{E_{fg,w}}{E_{ser}} > \frac{f_p}{f_r}$. For our implementations, the SRAM FPAA is lower energy when power drop-outs are at least 27-times less likely than fresh reconfigurations. However, a dedicated SPI port should reduce the serial transfer cost to E_{ser} ≈120nJ/column [231], which would favor SRAM FPAAs when no more than 1.2 power dropouts occur for each fresh reconfiguration.

For the system developer, knowledge of the reconfiguration energy is important for budgeting the overall system energy. The data in Table 11.3 can be used for this purpose. As an example, we can project the maximum acceptable frequency for reconfiguration of an SRAM FPAA based on the number of switches N_{sw}, the compression factor CF, and the percentage of reconfigurations that are initiated over-the-air (RX=%). The results are summarized in Table 11.4. We assume that the system should last 5 years on AA batteries, that only 5% of the system's energy budget is available for reconfiguration, and that compression is not used. For a 1k-switch FPAA, we can receive and program 2.6 configurations per hour. For a 100k-switch FPAA, the allowable frequency of configurations reduces to approximately once every two days, which illustrates the importance of compressing the configuration for large FPAAs. Excluding the communication costs, the allowable frequency of reconfiguration for a 1k-switch FPAA is over two per minute, which is sufficient to allow local adaptation of the FPAA settings.

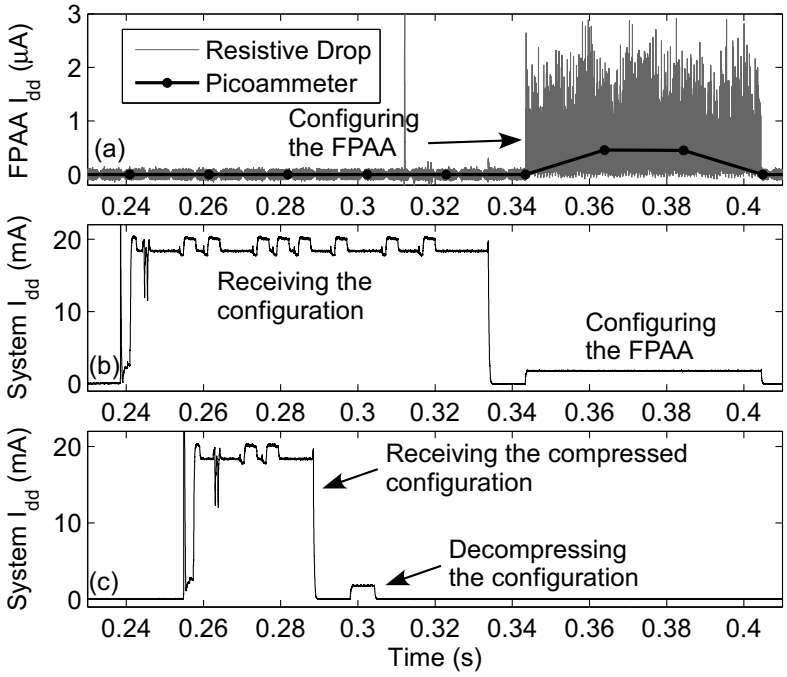

Figure 11.7: Measurements of reconfiguration energy for the 1280-switch 0.35μm SRAM-based FPAA. The supply currents were determined by measuring the voltage drop across suitably sized series resistors. (a) Supply current of the SRAM-based FPAA while being reconfigured. (b) Supply current of the sensor node while receiving and writing a configuration to an SRAM-based FPAA. (c) Supply current of the sensor node while receiving and decompressing a compressed FPAA configuration file.

Table 11.4: Maximum Frequency of Reconfiguration

	RX=100%			RX=10%
N_{sw}	CF=1	CF=3.3	CF=50	CF=1
10^3	2.6/hr	7.7/hr	33/hr	17/hr
10^4	6.3/day	18/day	3.3/hr	1.7/hr
10^5	0.63/day	1.8/day	8/day	4/day

11.5 Discussion of Reconfiguration Costs

In this Chapter, we have examined the costs of reconfiguring analog circuits in wireless sensors networks. We have investigated the sources of energy expenditure for both SRAM- and FG-based switch matrices and have introduced a new lower-energy mechanism for programming FG switches. SRAM-based switches are preferable in terms of speed, reconfiguration energy, device reliability, and scaling. However, FG switches offer lower reconfiguration costs when power drop-outs are more frequent than user-scheduled reconfigurations. Either way, our measurements of reconfiguration energy at the system level highlight the importance of optimizing node-to-node and IC-to-IC communications to reduce the total energy. We have shown that a simple method of compressing the FPAA configuration reduces the

total energy by 65%.

Chapter 12

Conclusions and Future Work

An infrastructure is currently being built around the world that will connect an unprecedented number of "physical" devices (i.e. devices with sensors and actuators). This infrastructure may eventually make the development of applications for networks of "physical" devices as simple as the development of web pages. Ultimately, the efficacy of this sensing infrastructure will depend upon the ability to handle data from diverse sensors at sustainable power levels. In this work, we have studied programmable analog signal processing as a solution to this data/power problem, and the results are very compelling.

This work has been guided by the notion that wireless sensor networks—because of the ultra-low energy budgets and the need to process sensor data locally—are a "killer app" for low-power analog signal processing (ASP). An "always-on" ASP can save power by monitoring sensor data while the network hibernates—hence our name for the paradigm, "Hibernets." Another appeal for applying ASP to wireless sensor networks is that the barrier to evaluating an ASP in a sensor network is relatively low, at least compared to the barrier for evaluating in medical implants, which are often cited as a target application for low-power ASP.

In the initial phase of this work, we focused on low-power analog processing circuits: a bandpass filter, a magnitude detector, and a hardware-based event detector. After initially applying these circuits to a wireless sensor network in an automobile-detection application, we found great potential for extending the battery life of sensor nodes. However, analog biasing was observed to be a major obstacle: first, the limited control over biasing was a constraint on the range of applications and usability of the ASP, and second, the power consumption of the biasing infrastructure was 40× larger than the power consumption of the processing circuitry. Although advancements in ASP circuits and algorithms were still needed, the significance of these advancements would have been questionable until the obstacles to analog biasing were overcome. As a result, much of this work has focused on the programmability of analog biases in sensor networks, although these contributions are valuable to all areas of analog signal processing.

The problem of programmable analog biasing is essentially the problem of analog memory, and the primary choice for nonvolatile analog memory in standard CMOS is the floating-gate transistor. While significant advancements have been made in analog floating-gate transistor research, these advancements have come from a small number of groups. As a result, a lot of knowledge about floating gates is experiential and word of mouth, which is a barrier for new

research. To address this barrier, we have studied the characterization, modeling, and design of floating-gate transistors, and have contributed several analytical and empirical findings, such as the optimal sizing of tunneling junctions, the tradeoffs between different types of tunneling junctions, and a step-by-step method for characterizing and parameterizing hot-electron injection current. These contributions remove a lot of "guess work" from the design of floating-gate transistors.

Prior methods for programming analog floating-gates have required too much overhead to be feasible for an ASP in a wireless sensor network. To solve this, we have investigated continuous-time programming techniques to develop a fast and efficient integrated programming system. Additionally, we have contributed the design of an integrated step-up converter for floating-gate programming, which is necessary for low-resource systems.

Our work on programmable analog biasing in wireless sensor networks has payed off, as embodied by our Netamorph 2.0 FPAA (field-programmable analog array), in which biasing is highly flexible and the power consumption of the biasing infrastructure has been reduced to a fraction of the power consumption of the processing circuitry. However, more obstacles exist. Primarily, the reliability of analog floating gates in standard CMOS is still an open question. We do have some anecdotal results: we have observed negligible shift in the center frequency of a bandpass filter during the first 48 hours after programming, and we have programmed a single floating gate 100k times and observed a relatively minor 50% reduction in programming speed. However, we still have questions about charge leakage over device lifetime and charge leakage in other processes. But our group now has much of the capabilities in place to answer many of these questions and results are expected to come soon.

After largely overcoming the obstacles to analog biasing in sensor networks, we repackaged the Hibernets processor into a reconfigurable architecture, and rebranded it "Netamorph." Joining the two research areas of FPAAs and wireless sensor networks led to contributions that would otherwise not be obvious, such as the importance of, as well as a method for, compressing FPAA configurations prior to transmission. Although we have made some contributions to the rigorous design of FPAA architectures, such as our studies of reconfiguration costs, average net lengths, and optimizing the number of terminals, much work is left to be done. Of great interest is the development of behavioral descriptions for FPAAs and tools that can decompose these descriptions into FPAA configurations. Such tools, in addition to making FPAAs easier to use, will aid in verifying our work on FPAA architecture tradeoffs, as well as enable quick exploration of new FPAA architectures and CAB elements.

Finally, whereas this work has focused on using analog signal processing for event detection to wake up a microcontroller, an alternate paradigm of using analog signal processing and digital signal processing in tandem may open many new possibilities, and is a line of research that is essentially unexplored.

References

[1] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 94–104, 1991.

[2] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A line in the sand: A wireless sensor network for target detection, classification, and tracking," *Computer Networks, Special Issue on Military Communications Systems and Technologies*, vol. 46, pp. 605–634, Dec. 2004.

[3] A. Arora, R. Ramnath, E. Ertin, P. Sinha, S. Bapat, V. Naik, V. Kulathumani, H. Zhang, H. Cao, M. Sridharan, S. Kumar, N. Seddon, C. Anderson, T. Herman, N. Trivedi, C. Zhang, R. Shah, S. Kulkarni, M. Aramugam, and L. Wang, "Exscal: Elements of an extreme scale wireless sensor network," in *Proceedings of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, Hong Kong, 2005, pp. 102–108.

[4] T. He, S. Krishnamurthy, J. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, G. Zhou, J. Hui, and B. Krogh, "Vigilnet: An integrated sensor network system for energy-efficient surveillance," *ACM Transactions on Sensor Networks*, vol. 2, pp. 1–38, Feb. 2006.

[5] M. Duarte and Y. Hu, "Vehicle classification in distributed sensor networks," *Journal of Parallel and Distributed Computing*, vol. 64, pp. 826–838, July 2004.

[6] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *Proceedings of the International Conference on Embedded Networked Sensor Systems*, Baltimore, MD, 2004, pp. 13–24.

[7] V. Raghunathan, C. Schurgers, S. Park, M. Srivastava, and B. Shaw, "Energy-aware wireless microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, pp. 40–50, March 2002.

[8] L. Nachman, J. Huang, J. Shahabdeen, R. Adler, and R. Kling, "Imote2: Serious computation at the edge," in *Proceedings of the International Wireless Communications and Mobile Computing Conference*, 2008, pp. 1118–1123.

[9] O. Berder and O. Sentieys, "PowWow: Power optimized hardware/software framework for wireless motes," in *International Conference on Architecture of Computing Systems*, 2010, pp. 1–5.

[10] K. Nose and T. Sakurai, "Optimization of VDD and VTH for low-power and high speed applications," in *Proceedings of the Asia and South Pacific Design Automation Conference*, 2000, pp. 469–474.

[11] J. Ko, K. Klues, C. Richter, W. Hofer, B. Kusy, M. Bruenig, T. Schmid, Q. Wang, P. Dutta, and A. Terzis, "Low power or high performance? A tradeoff whose time has come (and nearly gone)," in *Wireless Sensor Networks*. Springer, 2012, pp. 98–114.

[12] J. Hill and D. Culler, "Mica: A wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, 2002.

[13] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proceedings of the International Symposium on Information Processing in Sensor Networks*, Los Angeles, CA, 2005, pp. 364–369.

[14] R. Jurdak, K. Klues, B. Kusy, C. Richter, K. Langendoen, and M. Brunig, "Opal: A multiradio platform for high throughput wireless sensor networks," *IEEE Embedded Systems Letters*, vol. 3, no. 4, pp. 121–124, 2011.

[15] J. Lee, Y. Su, and C. Shen, "A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," in *Conference of the IEEE Industrial Electronics Society*, 2007, pp. 46–51.

[16] M. Jayalaksmi and K. Balasubramanian, "Simple capacitors to supercapacitors – an overview," *International Journal of Electrochemical Science*, pp. 1196–1217, Oct. 2008.

[17] L. Yerva, B. Campbell, A. Bansal, T. Schmid, and P. Dutta, "Grafting energy-harvesting leaves onto the sensornet tree," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, 2012, pp. 197–208.

[18] M. Gorlatova, J. Sarik, G. Grebla, M. Cong, I. Kymissis, and G. Zussman, "Movers and shakers: Kinetic energy harvesting for the Internet of Things," in *The 2014 ACM International Conference on Measurement and Modeling of Computer Systems*, 2014, pp. 407–419.

[19] S. Bandyopadhyay and A. Chandrakasan, "Platform architecture for solar, thermal, and vibration energy combining with MPPT and single inductor," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 9, pp. 2199–2215, Sept. 2012.

[20] S. Jevtic, M. Kotowsky, R. Dick, P. Dinda, and C. Dowding, "Lucid dreaming: Reliable analog event detection for energy-constrained applications," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, Cambridge, MA, 2007, pp. 350–359.

[21] M. Malinowski, M. Moskwa, M. Feldmeier, M. Laibowitz, and J. Paradiso, "Cargonet: A low-cost micropower sensor node exploiting quasi-passive wakeup for adaptive asynchronous monioring of exceptional events," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, Sydney, Australia, 2007, pp. 145–159.

[22] D. Goldberg, A. Andreou, P. Julian, P. Pouliquen, L. Riddle, and R. Rosasco, "A wake-up detector for an acoustic surveillance sensor network: Algorithm and VLSI implementation," in *Proceedings of the International Symposium on Information Processing in Sensor Networks*, Berkeley, CA, 2004, pp. 134–141.

[23] R. Dennard, F. Gaensslen, V. Rideout, E. Bassous, and A. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, 1974.

[24] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, "Scaling, power, and the future of CMOS," in *IEEE International Electron Devices Meeting*, 2005, pp. 7–15.

[25] B. Marr, B. Degnan, P. Hasler, and D. Anderson, "Scaling energy per operation via an asynchronous pipeline," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 21, no. 1, pp. 147–151, 2013.

[26] J. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of historical trends in the electrical efficiency of computing," *IEEE Annals of the History of Computing*, vol. 33, no. 3, pp. 46–54, 2011.

[27] R. Sarpeshkar, C. Salthouse, J.-J. Sit, M. Baker, S. Zhak, T.-T. Lu, L. Turicchia, and S. Balster, "An ultra-low-power programmable analog bionic ear processor," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 4, pp. 711–727, April 2005.

[28] R. Harrison and C. Charles, "A low-power low-noise CMOS amplifier for neural recording applications," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 6, pp. 958–965, June 2003.

[29] P. Hasler and D. Anderson, "Cooperative analog-digital signal processing," in *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, vol. 4, Orlando, FL, May 2002, pp. 3972–3975.

[30] R. Genov and G. Cauwenberghs, "Kerneltron: Support vector "machine" in silicon," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1426–1434, Sept. 2003.

[31] P. Smith, M. Kucic, R. Ellis, P. Hasler, and D. Anderson, "Mel-frequency cepstrum encoding in analog floating-gate circuitry," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2002.

[32] G. Cauwenberghs and V. Pedroni, "A low-power CMOS analog vector quantizer," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 1278–1283, 1997.

[33] J. Lazzaro, J. Wawrzynek, and R. Lippmann, "A micropower analog circuit implementation of hidden Markov model state decoding," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 1200–1209, 1997.

[34] R. Sarpeshkar, J. Kramer, G. Indiveri, and C. Koch, "Analog VLSI architectures for motion processing: From fundamental limits to system applications," *Proceedings of the IEEE*, vol. 84, pp. 969–987, 1996.

[35] M. Kucic, J. Dugger, P. Hasler, and D. Anderson, "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *Proceedings of the Conference on Advanced Research in VLSI*, Atlanta, GA, March 2001, pp. 148–162.

[36] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.

[37] ——, *Analog VLSI and neural systems*. Boston, MA, USA: Addison-Wesley, 1989.

[38] E. Vittoz, "Future of analog in the VLSI environment," in *IEEE International Symposium on Circuits and Systems*, vol. 2, May 1990, pp. 1372–1375.

[39] R. Sarpeshkar, "Analog versus digital: Extrapolating from electronics to neurobiology," *Neural Computation*, vol. 10, pp. 1601–1608, Oct. 1998.

[40] B. Murmann, "A/D converter trends: Power dissipation, scaling and digitally assisted architectures," in *IEEE Custom Integrated Circuits Conference*, Sept. 2008, pp. 105–112.

[41] B. Rumberg, D. Graham, V. Kulathumani, and R. Fernandez, "Hibernets: Energy-efficient sensor networks using analog signal processing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 3, pp. 321–334, Sept. 2011.

[42] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "Networked Slepian-Wolf: Theory, algorithms and scaling laws," *IEEE Transactions on Information Theory*, vol. 51, pp. 4057–4073, Dec. 2005.

[43] S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Processing Magazine*, vol. 19, pp. 51–60, March 2002.

[44] M. Gastpar, P. L. Dragotti, and M. Vetterli, "The Distributed Karhunen-Love Transform," *IEEE Transactions on Information Theory*, vol. 52, pp. 5177–5196, Dec. 2006.

[45] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *Proceedings of the International Conference on Data Engineering*, April 2006, pp. 3–7.

[46] R. Wagner, R. Baraniuk, S. Du, D. Johnson, and A. Cohen, "An architecture for distributed wavelet analysis and processing in sensor networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, Nashville, TN, 2006, pp. 243–250.

[47] J. Acimovic, B. Beferull-Lozano, and R. Cristescu, "Adaptive distributed algorithms for power-efficient data gathering in sensor networks," in *Proceedings of the International Conference on Wireless Networks, Communications and Mobile Computing*, vol. 2, 2005, pp. 946–951.

[48] A. Ciancio, S. Pattem, A. Ortega, and B. Krishnamachari, "Energy-efficient data representation and routing for wireless sensor networks based on a distributed wavelet compression algorithm," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, Nashville, TN, 2006, pp. 309–316.

[49] G. Shen and A. Ortega, "Joint routing and 2D transform optimization for irregular sensor network grids using wavelet lifting," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, St. Louis, MO, 2008, pp. 183–194.

[50] S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 4, pp. 60–66, Sept. 2008.

[51] S. Pattem, G. Shen, Y. Chen, B. Krishnamachari, and A. Ortega, "SenZip: An architecture for distributed en-route compression in wireless sensor networks," in *Workshop on Sensor Networks for Earth and Space Science Applications*, April 2009.

[52] P. von Rickenbach and R. Wattenhofer, "Gathering correlated data in sensor networks," in *Proceedings of the Joint Workshop on Foundations of Mobile Computing*, Philadelphia, PA, 2004, pp. 60–66.

[53] G. Shen, S. Pattem, and A. Ortega, "Energy-efficient graph-based wavelets for distributed coding in wireless sensor networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2009, pp. 2253–2256.

[54] R. Cristescu, B. Beferull-Lozano, M. Vetterli, and R. Wattenhofer, "Network correlated data gathering with explicit communication: NP-completeness and algorithms," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 41–54, Feb. 2006.

[55] S. Servetto, "Sensing LENA - massively distributed compression of sensor images," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, 2003, pp. 613–616.

[56] J. Gao, L. Guibas, N. Milosavljivec, and J. Hershberger, "Sparse data aggregation in sensor networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, Cambridge, MA, 2007, pp. 430–439.

[57] R. Sarkar, X. Zhu, and J. Gao, "Hierarchical spatial gossip for multi-resolution representations in sensor networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, Cambridge, MA, 2007, pp. 420–429.

[58] V. Kulathumani and A. Arora, "Distance sensitive snapshots in wireless sensor networks," in *Proceedings of the International Conference on Principles of Distributed Systems*, Guadeloupe, French West Indies, 2007, pp. 143–158.

[59] P. Hasler, P. Smith, D. Graham, R. Ellis, and D. Anderson, "Analog floating-gate, on-chip auditory sensing system interfaces," *IEEE Sensors Journal*, vol. 5, pp. 1027–1034, Oct. 2005.

[60] B. Greenstein, C. Mar, A. Pesterev, S. Farshchi, E. Kohler, J. Judy, and D. Estrin, "Capturing high-frequency phenomena using a bandwidth-limited sensor network," in *Proceedings of the International Conference on Embedded Networked Sensor Systems*, Boulder, CO, 2006, pp. 279–292.

[61] C. T. Inc., "Stargate Gateway (SPB400)," http://www.willow.co.uk/.

[62] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *Architectural Support for Programming Languages and Operating Systems*, vol. 35, Nov. 2000, pp. 93–104.

[63] J. Frigo, V. Kulathumani, S. Brennan, and E. Raby, "Sensor network based vehicle classification and license plate identification system," in *Proceedings of the International Conference on Networked Sensing Systems*, Pittsburgh, PA, 2009, pp. 224–227.

[64] B. Malhotra, I. Nikolaidis, and J. Harms, "A simple vehicle classification framework for wireless audio-sensor networks," *Journal of Telecommunications and Information Technology*, pp. 43–50, Jan. 2008.

[65] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a wireless sensor network platform for detecting rare, random, and ephemeral events," in *Proceedings of the International Symposium on Information Processing in Sensor Networks*, Los Angeles, CA, 2006, pp. 497–502.

[66] V. Ekanayake, C. Kelly IV, and R. Manohar, "An ultra low-power processor for sensor networks," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, Boston, MA, 2004, pp. 27–36.

[67] B. Zhai, S. Pant, L. Nazhandali, S. Hanson, J. Olson, A. Reeves, M. Minuth, R. Helfand, T. Austin, D. Sylvester, and D. Blaauw, "Energy-efficient subthreshold processor design," *IEEE Transactions on Very Large Scale Integrated Systems*, vol. 17, pp. 1127–1137, Aug. 2009.

[68] M. Sheets, F. Burghardt, T. Karalar, J. Ammer, Y. Chee, and J. Rabaey, "A power-managed protocol processor for wireless sensor networks," in *Proceedings of the IEEE Symposium on VLSI Circuits*, 2006, pp. 262–263.

[69] M. Hempstead, N. Tripathi, P. Mauro, G.-Y. Wei, and D. Brooks, "An ultra low power system architecture for sensor network applications," in *Proceedings of the International Symposium on Computer Architecture*, 2005, pp. 208–219.

[70] B. Linares-Barranco, E. Sanchez-Sinencio, A. Rodriguez-Vazquez, and J. Huertas, "A CMOS analog adaptive BAM with on-chip learning and weight refreshing," *IEEE Transactions on Neural Networks*, vol. 4, pp. 445–455, 1993.

[71] H. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarkoy, "Probability propagation and decoding in analog VLSI," *IEEE Transactions on Information Theory*, vol. 47, pp. 837–843, 2001.

[72] T. Hall, C. Twigg, P. Hasler, and D. Anderson, "Application performance of elements in a floating-gate FPAA," in *Proceedings of the International Symposium on Circuits and Systems*, vol. 2, Vancouver, Canada, 2004, pp. 589–592.

[73] Y. Taur, "CMOS design near the limit of scaling," *IBM Journal of Research and Development*, vol. 46, pp. 213–222, 2002.

[74] A. van Schaik, E. Fragnière, and E. Vittoz, "Improved silicon cochlea using compatible lateral bipolar transistors," in *Advances in Neural Information Processing Systems 8*, Cambridge, MA, 1996, pp. 671–677.

[75] R. Sarpeshkar, R. Lyon, and C. Mead, "An analog VLSI cochlea with new transconductance amplifiers and nonlinear gain control," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, Atlanta, GA, 1996, pp. 292–296.

[76] D. Graham and P. Hasler, "Capacitively-coupled current conveyer second-order sections for continuous-time bandpass filtering and cochlea modeling," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, Scottsdale, AZ, May 2002, pp. 485–488.

[77] *American National Standard Specification for Octave-Band and Fractional-Octave-Band Analog and Digital Filter*, ANSI S1.11-1986 ed., American National Standards Institute.

[78] B. Rumberg, D. Graham, and V. Kulathumani, "Hibernets: Energy-efficient sensor networks using analog signal processing," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, Stockholm, Sweden, 2010, pp. 129–139.

[79] D. Graham, P. Hasler, R. Chawla, and P. Smith, "A low-power, programmable bandpass filter section for higher-order filter applications," *IEEE Transactions on Circuits and Systems I*, vol. 54, no. 6, pp. 1165–1176, June 2007.

[80] J. Rice, K. Mechitov, F. Spencer Jr., and G. Agha, "Autonomous smart sensor network for full-scale structural health monitoring," in *Proceedings of the SPIE Conference on Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*, vol. 7647, 2010.

[81] W. Hu, V. Tran, N. Bulusu, C. Chou, S. Jha, and A. Taylor, "The design and evaluation of a hybrid sensor network for cane-toad monitoring," in *Proceedings of the International Symposium on Information Processing in Sensor Networks*, 2005.

[82] R. Edwards and G. Cauwenberghs, "Mixed-mode correlator for micropower acoustic transient classification," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 10, pp. 1367–1372, Oct. 1999.

[83] T. Yamasaki and T. Shibata, "Analog soft-pattern-matching classifier using floating-gate MOS technology," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1257–1265, Sept. 2003.

[84] S. Peng, P. Hasler, and D. Anderson, "An analog programmable multidimensional radial basis function based classifier," *IEEE Transactions on Circuits and Systems I*, vol. 54, no. 10, pp. 2148–2158, Oct 2007.

[85] T. Hall, C. Twigg, J. Gray, P. Hasler, and D. Anderson, "Large-scale field-programmable analog arrays for analog signal processing," *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 11, pp. 2298–2307, Nov. 2005.

[86] B. Rumberg and D. Graham, "A low-power and high-precision programmable analog filter bank," *IEEE Transactions on Circuits Systems II*, vol. 59, no. 4, pp. 234–238, April 2012.

[87] G. Strang and T. Nguyen, *Wavelets and filter banks*. Wellesley-Cambridge Press, 1996.

[88] M. Kucic, A. Low, P. Hasler, and J. Neff, "A programmable continuous-time floating-gate Fourier processor," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 1, pp. 90–99, Jan. 2001.

[89] W. Liu, M. Goldstein, Jr., and A. Andreou, "Multiresolution speech analysis with an analog cochlear model," in *Proceedings of the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, Victoria, BC, Oct. 1992, pp. 433–436.

[90] B. Wen and K. Boahen, "A 360-channel speech preprocessor that emulates the cochlear amplifier," in *IEEE ISSCC Digest of Technical Papers*, Feb. 2006, pp. 2268–2277.

[91] E. Fragnière, "A 100-channel analog CMOS auditory filter bank for speech recognition," in *IEEE ISSCC Digest of Technical Papers*, vol. 1, Feb. 2005, pp. 140–141.

[92] S.-C. Liu, A. van Schaik, B. Minch, and T. Delbrück, "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms," in *Proceedings of IEEE International Symposium on Circuits and Systems*, May 2010, pp. 2027–2030.

[93] R. Sarpeshkar, R. Lyon, and C. Mead, "A low-power wide-dynamic-range analog VLSI cochlea," *Analog Integrated Circuits and Signal Processing*, vol. 16, no. 3, pp. 245–274, 1998.

[94] A. Katsiamis, E. Drakakis, and R. Lyon, "A biomimetic, 4.5 μw, 120+ dB, log-domain cochlea channel with AGC," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 3, pp. 1006–1022, March 2009.

[95] J. Georgiou and C. Toumazou, "A 126-μw cochlear chip for a totally implantable system," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 2, pp. 430–443, Feb. 2005.

[96] M. Baker, T. Lu, C. Salthouse, J. Sit, S. Zhak, and R. Sarpeshkar, "A 16-channel analog VLSI processor for bionic ears and speech-recognition front ends," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 2003, pp. 521–526.

[97] D. Graham, P. Smith, R. Chawla, and P. Hasler, "A programmable bandpass array using floating-gate elements," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, Vancouver, BC, Canada, May 2004, pp. I–97–100.

[98] R. Sarpeshkar, R. Lyon, and C. Mead, "A low-power wide-linear-range transconductance amplifier," *Analog Integrated Circuits and Signal Processing*, vol. 13, pp. 123–151, 1997.

[99] P. Furth and H. Ommani, "Low-voltage highly-linear transconductor design in subthreshold CMOS," in *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, vol. 1, Sacramento, CA, Aug. 1997, pp. 156–159.

[100] C. Salthouse and R. Sarpeshkar, "A practical micropower programmable bandpass filter for use in bionic ears," *IEEE Journal of Solid State Circuits*, vol. 38, no. 1, pp. 63–70, Jan. 2003.

[101] K. Odame, D. Anderson, and P. Hasler, "A bandpass filter with inherent gain adaptation for hearing applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 3, pp. 786–795, April 2008.

[102] O. Omeni, E. Rodríguez-Villegas, and C. Toumazou, "A micropower CMOS continuous-time filter with on-chip automatic tuning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 4, pp. 695–705, April 2005.

[103] L. Pylarinos and K. Phang, "Low-voltage programmable g_m-C filter for hearing aids using dynamic gate biasing," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, May 2005, pp. 1984–1987.

[104] E. Rodriguez-Villegas, A. Yùfera, and A. Rueda, "A 1.25-V micropower G_m-C filter based on FGMOS transistors operating in weak inversion," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 1, pp. 100–111, Jan. 2004.

[105] A. Bandyopadhyay, G. Serrano, and P. Hasler, "Adaptive algorithm using hot-electron injection for programming analog computational memory elements within 0.2% of accuracy over 3.5 decades," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 9, pp. 2107–2114, Sept. 2006.

[106] B. Rumberg and D. Graham, "A low-power magnitude detector for analysis of transient-rich signals," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 3, pp. 676–685, March 2012.

[107] F. Yuan, "Design techniques for ASK demodulators of passive wireless microsystems: a state-of-the-art review," *Analog Integrated Circuits and Signal Processing*, vol. 63, pp. 33–45, 2010.

[108] M. Baker and R. Sarpeshkar, "Low-power single-loop and dual-loop AGCs for bionic ears," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 9, pp. 1983–1996, Sept. 2006.

[109] J. Picone, "Signal modeling techniques in speech recognition," *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1215–1247, Sept. 1993.

[110] J. Alegre, S. Celma, B. Calvo, and J. García del Pozo, "A novel CMOS envelope detector structure," *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2007.

[111] X. Arreguit and E. Vittoz, "Perception systems implemented in analog VLSI for real-time applications," in *Proceedings of the Perception to Action Conference*, Lausanne, Switzerland, 1994, pp. 170–180.

[112] R. Edwards and G. Cauwenberghs, "Log-domain circuits for auditory signal processing," in *IEEE Midwest Symposium on Circuits and Systems*, vol. 2, Aug. 1999, pp. 968–971.

[113] S. Haddad, J. Karel, R. Peelers, R. Westra, and W. Serdijn, "Ultra low-power analog Morlet wavelet filter in 0.18μm BiCMOS technology," in *Proceedings of the European Solid-State Circuits Conference*, Sept. 2005, pp. 323–326.

[114] R. Ellis, H. Yoo, D. Graham, P. Hasler, and D. Anderson, "A continuous-time speech enhancement front-end for microphone inputs," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, May 2002, pp. II–728–731.

[115] S. Ravindran, C. Demiroglu, and D. Anderson, "Speech recognition using filter-bank features," in *Proceedings of the IEEE Asilomar Conference on Signals, Systems, and Computers*, vol. 2, Pacific Grove, CA, Nov. 2003, pp. 1900–1903.

[116] H. Abdalla and T. Horiuchi, "An analog VLSI low-power envelope periodicity detector," *IEEE Transactions on Circuits and Systems I, Regular Papers*, vol. 52, no. 9, pp. 1709–1720, Sept. 2005.

[117] S.-B. Park, J. Wilson, and M. Ismail, "Peak detectors for multistandard wireless receivers," *IEEE Circuits and Devices Magazine*, vol. 22, no. 6, pp. 6–9, Nov.-Dec. 2006.

[118] M. Kruiskamp and D. Leenaerts, "A CMOS peak detect sample and hold circuit," *IEEE Transactions on Nuclear Science*, vol. 41, no. 1, pp. 295–298, Feb. 1994.

[119] J. Taylor, "Describing functions," *Electrical and Electronics Engineering Encyclopedia*, vol. Supplement 1, pp. 77–98, 2000.

[120] R. Gilmore and M. Steer, "Nonlinear circuit analysis using the method of harmonic balance – A review of the art. Part I. Introductory concepts," *International Journal of Microwave and Millimeter-Wave Computer-Aided Engineering*, vol. 1, no. 1, pp. 22–37, 1991.

[121] T. Delbrück, ""Bump" circuits for computing similarity and dissimilarity of analog voltages," in *Proceedings of the International Joint Conference on Neural Networks*, Seattle, WA, USA, July 1991, pp. I–475–479.

[122] M. Cohen and A. Andreou, "MOS circuit for nonlinear Hebbian learning," *Electronics Letters*, vol. 28, no. 9, pp. 809–810, 1992.

[123] A. Gelb and W. Vander Velde, *Multiple-input describing functions and nonlinear system design.* New York: McGraw-Hill, 1968.

[124] S. Zhak, M. Baker, and R. Sarpeshkar, "A low-power wide dynamic range envelope detector," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 10, pp. 1750–1753, Oct. 2003.

[125] M. Steyaert, W. Dehaene, J. Craninckx, M. Walsh, and P. Real, "A CMOS rectifier-integrator for amplitude detection in hard disk servo loops," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 7, pp. 743–751, July 1995.

[126] Y. Zhou, G. Huang, S. Nam, and B.-S. Kim, "A novel wide-band envelope detector," in *Proceedings of the IEEE Radio Frequency Integrated Circuits Symposium*, June 2008, pp. 219–222.

[127] E. Rodriguez-Villegas, P. Corbishley, C. Lujan-Martinez, and T. Sanchez-Rodriguez, "An ultra-low-power precision rectifier for biomedical sensors interfacing," *Sensors and Actuators A: Physical*, vol. 153, no. 2, pp. 222–229, 2009.

[128] J. Alegre, S. Celma, J. García del Pozo, and N. Medrano, "Fast-response low-ripple envelope follower," *Integration, the VLSI Journal*, vol. 42, no. 2, pp. 169–174, 2009.

[129] P. Sarkar and S. Chakrabartty, "Compressive self-powering of piezo-floating-gate mechanical impact detectors," *IEEE Transactions of Circuits and Systems – I*, vol. 60, no. 9, Sept. 2013.

[130] Y. Wong, M. Cohen, and P. Abshire, "A 750-MHz 6-b adaptive floating-gate quantizer in 0.35-μm CMOS," *IEEE Transactions on Circuits and Systems—I*, vol. 56, pp. 1301–1312, July 2009.

[131] B. Rumberg and D. Graham, "A floating-gate memory cell for continuous-time programming," in *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, Boise, ID, August 2012, pp. 214–217.

[132] P. Hasler, B. Minch, and C. Diorio, "Floating-gate devices: They are not just for digital memories anymore," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1999, pp. 388–391.

[133] P. Pavan, R. Bez, P. Olivo, and E. Zanoni, "Flash memory cells—an overview," *Proceedings of the IEEE*, vol. 85, no. 8, pp. 1248–1271, 1997.

[134] Y. Hu and P. Georgiou, "A robust ISFET pH-measuring front-end for chemical reaction monitoring," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 2, pp. 177–185, April 2014.

[135] B. Minch, P. Hasler, and C. Diorio, "Multiple-input translinear element networks," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 1, pp. 20–28, 2001.

[136] C. Diorio, "A p-channel MOS synapse transistor with self-convergent memory writes," *IEEE Transactions on Electron Devices*, vol. 47, no. 2, pp. 464–472, Feb. 2000.

[137] S. Song, K. Chun, and C. Kim, "A logic-compatible embedded flash memory for zero-standby power system-on-chips featuring a multi-story high voltage switch and a selective refresh scheme," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 5, pp. 1302–1314, 2013.

[138] C. Huang, P. Sarkar, and S. Chakrabartty, "Rail-to-rail, linear hot-electron injection programming of floating-gate voltage bias generators at 13-bit resolution," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 11, pp. 2685–2692, Nov. 2011.

[139] C. Diorio, S. Mahajan, P. Hasler, B. Minch, and C. Mead, "A high-resolution nonvolatile analog memory cell," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, Seattle, WA, April 1995, pp. 2233–2236.

[140] K.-H. Kim, K. Lee, T.-S. Jung, and K.-D. Suh, "An 8-bit-resolution, 360-μs write time nonvolatile analog memory based on differentially balanced constant-tunneling-current scheme (DBCS)," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 11, pp. 1758–1762, Nov. 1998.

[141] H. Román and G. Serrano, "A system architecture for automated charge modifications of analog memories," in *IEEE Midwest Symp. Circuits Syst.*, Aug. 2010, pp. 1069–1072.

[142] Y.-D. Wu, K.-C. Cheng, C.-C. Lu, and H. Chen, "Embedded analog nonvolatile memory with bidirectional and linear programmability," *IEEE Transactions on Circuits and Systems–II: Express Briefs*, vol. 59, no. 2, pp. 88–92, Feb. 2012.

[143] A. Andreou, K. Boahen, P. Pouliquen, A. Pavasović, R. Jenkins, and K. Strohbehn, "Current-mode subthreshold MOS circuits for analog VLSI neural systems," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 205–213, March 1991.

[144] S. Rapp, K. McMillan, and D. Graham, "Spice-compatible modelling technique for simulating floating-gate transistors," *Electronics Letters*, vol. 47, no. 8, pp. 483–485, April 2011.

[145] B. Rumberg and D. Graham, "Efficiency and reliability of Fowler-Nordheim tunnelling in CMOS floating-gate transistors," *Electronics Letters*, vol. 49, no. 23, pp. 1484–1486, 2013.

[146] L. Carley, "Trimming analog circuits using floating-gate analog MOS memory," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 6, pp. 1569–1575, 1989.

[147] M. Lenzlinger and E. Snow, "Fowler-Nordheim tunneling into thermally grown SiO2," *Journal of Applied Physics*, vol. 40, no. 1, pp. 278–283, 1969.

[148] Z. Weinberg, "On tunneling in metal-oxide-silicon structures," *Journal of Applied Physics*, vol. 53, no. 7, pp. 5052–5056, 1982.

[149] P. Hasler, B. Minch, and C. Diorio, "Adaptive circuits using pFET floating-gate devices," in *Proc. IEEE ARVLSI*, 1999, pp. 215–229.

[150] K. Yang, Y. King, and C. Hu, "Quantum effect in oxide thickness determination from capacitance measurement," in *Proc. IEEE VLSIT*, 1999, pp. 77–78.

[151] A. Kolodny, S. Nieh, B. Eitan, and J. Shappir, "Analysis and modeling of floating-gate EEPROM cells," *IEEE Transactions on Electron Devices*, vol. 33, no. 6, pp. 835–844, 1986.

[152] Y. Park and D. Schroder, "Degradation of thin tunnel gate oxide under constant Fowler-Nordheim current stress for a Flash EEPROM," *IEEE Transactions on Electron Devices*, vol. 45, no. 6, pp. 1361–1368, 1998.

[153] B. Streetman and S. Banerjee, *Solid state electronic devices*. Prentice Hall, 2006.

[154] T. Ong, P. Ko, and C. Hu, "Hot-carrier current modeling and device degradation in surface-channel p-MOSFETs," *IEEE Transactions on Electron Devices*, vol. 37, no. 7, pp. 1658–1666, 1990.

[155] P. Hasler, A. Andreou, C. Diorio, B. Minch, and C. Mead, "Impact ionization and hot-electron injection derived consistently from Boltzmann transport," *VLSI Design*, vol. 8, no. 1-4, pp. 454–461, 1998.

[156] W. Shockley, "Problems related to p-n junctions in silicon," *Solid-State Electronics*, vol. 2, pp. 35–67, 1961.

[157] C. Hu, "Lucky-electron model of channel hot electron emission," in *International Electron Devices Meeting*, vol. 25, 1979, pp. 22–25.

[158] S. Tam, P. Ko, and C. Hu, "Lucky-electron model of channel hot-electron injection in MOSFET's," *IEEE Transactions on Electron Devices*, vol. 31, no. 9, pp. 1116–1125, 1984.

[159] P. Hasler, A. Basu, and S. Kozil, "Above threshold pFET injection modeling intended for programming floating-gate systems," in *IEEE International Symposium on Circuits and Systems*, 2007, pp. 1557–1560.

[160] J. Chung, M. Jeng, G. May, P. Ko, and C. Hu, "Hot-electron currents in deep-submicrometer MOSFETs," in *International Electron Devices Meeting*, 1988, pp. 200–203.

[161] V. Srinivasan, D. Graham, and P. Hasler, "Floating-gates transistors for precision analog circuit design: an overview," in *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, vol. 1, Covington, KY, Aug. 2005, pp. 71–74.

[162] The MOSIS Service, *Wafer Electrical Test Data and SPICE Model Parameters*. http://www.mosis.com/requests/test-data, 2014.

[163] International Technology Roadmap for Semiconductors, *Process Integration, Devices, and Structures*. http://www.itrs.net/Links/2013ITRS/Home2013.htm, 2013.

[164] J. Starzyk, Y. Jan, and F. Qiu, "A DC-DC charge pump design based on voltage doublers," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 3, pp. 350–359, 2001.

[165] O. Wong, H. Wong, W. Tam, and C. Kok, "A comparative study of charge pumping circuits for Flash memory applications," *Microelectronics Reliability*, vol. 52, pp. 670–687, 2012.

[166] J. Cockcroft and E. Walton, "Experiments with high velocity positive ions. (I) further developments in the method of obtaining high velocity positive ions," *Royal Society of London Proceedings Series A*, vol. 136, pp. 619–630, 1932.

[167] J. Dickson, "On-chip high-voltage generation in MNOS integrated circuits using an improved voltage multiplier technique," *IEEE Journal of Solid-State Circuits*, vol. 11, no. 3, pp. 374–378, 1976.

[168] G. Palumbo and D. Pappalardo, "Charge pump circuits: An overview on design strategies and topologies," *IEEE Circuits and Systems Magazine*, vol. 10, no. 1, pp. 31–45, 2010.

[169] B. Gregoire, "A compact switched-capacitor regulated charge pump power supply," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 8, pp. 1944–1953, 2006.

[170] C. Wu and C. Chen, "A low-ripple charge pump with continuous pumping current control," in *IEEE Midwest Symposium on Circuits and Systems*, 2008, pp. 722–725.

[171] Y. Kang *et al*, "High-voltage analog system for a mobile NAND flash," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 507–517, 2008.

[172] E. Bayer and H. Schmeller, "Charge pump with active cycle regulation—closing the gap between linear and skip modes," in *IEEE Power Electronics Specialists Conference*, vol. 3, 2000, pp. 1497–1502.

[173] J. Lee, S. Kim, S. Song, J. Kim, S. Kim, and H. Yoo, "A regulated charge pump with small ripple voltage and fast start-up," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 2, pp. 425–432, 2006.

[174] L. Aaltonen and K. Halonen, "On-chip charge-pump with continuous frequency regulation for precision high-voltage generation," in *Ph.D. Research in Microelectronics and Electronics (PRIME)*, San Francisco, CA, July 2009, pp. 68–71.

[175] M. Kuriyama, S. Atsumi, A. Umezawa, H. Banba, K. Imamiya, K. Naruke, S. Yamada, E. Obi, M. Oshikiri, T. Suzuki, M. Wada, and S. Tanaka, "A 5V-only 0.6μm Flash EEPROM with row decoder scheme in triple-well structure," in *IEEE Solid-State Circuits Conference*, San Francisco, CA, Feb. 1992, pp. 152–153.

[176] N. Li, Z. Huang, M. Jiang, and Y. Inoue, "High efficiency four-phase all PMOS charge pump without body effect," in *International Conference on Communications, Circuits and Systems*, May 2008, pp. 1083–1087.

[177] J. Shin, I. Chung, Y. Park, and H. Min, "A new charge pump without degradation in threshold voltage due to body effect," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 8, pp. 1227–1230, Aug. 2000.

[178] H. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, vol. 19, no. 4, pp. 468–473, 1984.

[179] A. Chandrakasan and R. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proceedings of the IEEE*, vol. 83, no. 4, pp. 498–523, 1995.

[180] C. Yoo, "A CMOS buffer without short-circuit power consumption," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 9, pp. 935–937, 2000.

[181] C. Lee and P. Mok, "A monolithic current-mode CMOS DC-DC converter with on-chip current-sensing technique," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 1, pp. 3–14, 2004.

[182] B. Rumberg, D. Graham, and V. Kulathumani, "A low-power, programmable analog event detector for resource-constrained sensing systems," in *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, Boise, ID, August 2012, pp. 338–341.

[183] P. Furth and A. Andreou, "Linearised differential transconductors in subthreshold CMOS," *Electron. Lett.*, pp. 545–547, 1995.

[184] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, pp. 21–45, 2006.

[185] L. Itti and C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention," *Vision Research*, vol. 40, pp. 1489–1506, 2000.

[186] Y. Zhai and M. Shah, "Visual attention detection in video sequences using spatiotemporal cues," in *ACM International Conference on Multimedia*, 2006, pp. 815–824.

[187] O. Kalinli and S. Narayanan, "A saliency-based auditory attention model with applications to unsupervised prominent syllable detection in speech," in *INTERSPEECH*, 2007, pp. 1941–1944.

[188] L. Itti, "Automatic foveation for video compression using a neurobiological model of visual attention," *IEEE Transactions on Image Processing*, vol. 13, no. 10, pp. 1304–1318, Oct. 2004.

[189] C. Diorio, P. Hasler, B. Minch, and C. Mead, "A floating-gate MOS learning array with locally computed weight updates," *IEEE Transactions on Electron Devices*, vol. 44, no. 12, pp. 2281–2289, Dec. 1997.

[190] B. Gestner, J. Tanner, and D. Anderson, "Glass break detector analog front-end using novel classifier circuit," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 3586–3589.

[191] T. Delbrück, T. Koch, R. Berner, and H. Hermansky, "Fully integrated 500μW speech detection wake-up circuit," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, May 2010, pp. 2015–2018.

[192] B. Kelly, B. Rumberg, D. Graham, and V. Kulathumani, "Reconfigurable analog signal processing for wireless sensor networks," in *Proceedings of the International Midwest Symposium on Circuits and Systems*, Columbus, OH, USA, Aug. 2013, pp. 221–224.

[193] B. Rumberg, B. Kelly, D. Graham, and V. Kulathumani, "Demo Abstract: Netamorph: Field-programmable analog arrays for energy-efficient sensor networks," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, Philadelphia, PA, USA, April 2013, pp. 309–310.

[194] S. N. Pakzad, G. L. Fenves, S. Kim, and D. E. Culler, "Design and implementation of scalable wireless sensor network for structural monitoring," *Journal of Infrastructure Systems*, vol. 14, no. 1, pp. 89–101, 2008.

[195] D. Anderson *et al*, "A field programmable analog array and its application," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, Aug. 1997, pp. 555–558.

[196] B. Pankiewicz, M. Wojcikowski, S. Szczepanski, and Y. Sun, "A field programmable analog array for CMOS continuous-time OTA-C filter applications," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 2, pp. 125–136, Feb. 2002.

[197] E. Mackensen and C. Müller, "Implementation of reconfigurable micro-sensor interfaces utilizing FPAAs," in *IEEE Sensors*, Oct. 2005, pp. 1064–1067.

[198] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. Twigg, and P. Hasler, "A floating-gate-based field-programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 9, pp. 904–922, Sept. 2010.

[199] D. Fernández, L. Martínez-Alvarado, and J. Madrenas, "A translinear, log-domain FPAA on standard CMOS technology," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 2, pp. 490–503, Feb. 2012.

[200] I. Kuon, R. Tessier, and J. Rose, "Fpga architecture: Survey and challenges," *Foundations and Trends in Electronic Design Automation*, vol. 2, no. 2, pp. 135–253, 2008.

[201] E. Lee and P. Gulak, "A CMOS field-programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 1860–1867, Dec. 1991.

[202] C. Schlottmann, S. Shapero, S. Nease, and P. Hasler, "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 9, pp. 2174–2184, Sept. 2012.

[203] R. Harrison and C. Koch, "A robust analog VLSI Reichardt motion sensor," *Analog Integrated Circuits and Signal Processing*, vol. 24, pp. 213–229, 2000.

[204] O. Schwartz and E. Simoncelli, "Natural signal statistics and sensory gain control," *Nature Neuroscience*, vol. 4, pp. 819–825, 2001.

[205] B. Rumberg and D. Graham, "Reconfiguration costs in analog sensor interfaces for wireless sensing applications," in *Proceedings of the International Midwest Symposium on Circuits and Systems*, Columbus, OH, USA, Aug. 2013, pp. 321–324.

[206] D. D'Mello and G. Gulak, "Design approaches to field-programmable analog integrated circuits," *Analog Integrated Circuits and Signal Processing*, no. 17, pp. 7–34, June 1998.

[207] F. Mims, *The Forrest Mims Circuit Scrapbook, Volume 2.* Eagle Rock, VA, USA: LLH Technology Publishing, 2000.

[208] Analog Devices, "Monolithic peak detector with reset and hold mode," in *PDK01 datasheet*, 2001.

[209] M. Sivilotti, "A dynamically configurable architecture for prototyping analog circuits," in *Proceedings of the Fifth MIT Conference on Advanced Research in VLSI*, Cambridge, MA, USA, 1988, pp. 237–258.

[210] D. Vallancourt and Y. Tsividis, "Timing-controlled fully programmable analogue signal processors using switched continuous-time filters," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 8, pp. 947–954, 1988.

[211] E. Lee and G. Gulak, "MOS transconductor-based field programmable analog array," in *3rd International Workshop on Post-Binary ULSI Systems*, Boston, MA, USA, 1994.

[212] P. Chow and G. Gulak, "A field-programmable mixed-analog-digital array," in *Proceedings of the 1995 ACM Third International Symposium on Field-programmable Gate Arrays*, 1995, pp. 104–109.

[213] E. Pierzchala, M. Perkowski, P. V. Halen, and R. Schaumann, "Current-mode amplifier/integrator for a field-programmable analog array," in *IEEE International Solid-State Circuits Conference*, 1995, pp. 196–197.

[214] H. Kutuk and S. Kang, "A field-programmable analog array (FPAA) using switched-capacitor techniques," in *IEEE International Symposium on Circuits and Systems*, vol. 4, 1996, pp. 41–44.

[215] S. Chang, B. Hayes-Gill, and C. Paull, "Multi-function block for a switched current field programmable analogue array," in *IEEE Midwest Symposium on Circuits and Systems*, 1996, pp. 158–161.

[216] V. Gaudet and G. Gulak, "CMOS implementation of a current conveyor-based field-programmable analog array," in *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems, and Computers*, vol. 2, 1997, pp. 1156–1159.

[217] X. Quan, S. Embabi, and E. Sanchez-Sinencio, "A current-mode based field programmable analog array architecture for signal processing applications," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 1998, pp. 277–280.

[218] M. Mar, B. Sullam, and E. Blom, "An architecture for a configurable mixed-signal device," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 3, pp. 565–568, 2003.

[219] D. Varghese and J. Ross, "A continuous-time hierarchical field programmable analogue array for rapid prototyping and hierarchical approach to analogue systems design," in *Symposium on Integrated Circuits and Systems Design*, 2005, pp. 248–253.

[220] C. Twigg and P. Hasler, "A large-scale reconfigurable analog signal processor (RASP) IC," in *IEEE Custom Integrated Circuits Conference*, 2006, pp. 5–8.

[221] P. Hasler and C. Twigg, "An OTA-based large-scale field programmable analog array (FPAA) for faster on-chip communication and computation," in *IEEE International Symposium on Circuits and Systems*, 2007, pp. 177–180.

[222] J. Becker, F. Henrici, S. Trendelenburg, M. Ortmanns, and Y. Manoli, "A field-programmable analog array of 55 digitally tunable OTAs in a hexagonal lattice," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 12, pp. 2759–2768, Dec. 2008.

[223] S. Peng, G. Gurun, C. Twigg, M. Qureshi, A. Basu, S. Brink, P. Hasler, and F. Degertekin, "A large-scale reconfigurable smart sensory chip," in *IEEE International Symposium on Circuits and Systems*, 2009, pp. 2145–2148.

[224] P. Lajevardi, A. Chandrakasan, and H. Lee, "Zero-crossing detector based reconfigurable analog system," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 11, pp. 2478–2487, 2011.

[225] R. Wunderlich, F. A. P. and Hasler, "Floating gate-based field programmable mixed-signal array," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 8, pp. 1496–1505, 2013.

[226] X. Cheng, H. Yang, T. Yin, Q. Wu, T. Zhi, and F. Liu, "Mixed-grained CMOS field programmable analog array for smart sensory applications," *Journal of Electronics (China)*, vol. 31, no. 2, pp. 129–142, 2014.

[227] P. Christie and D. Stroobandt, "The interpretation and application of Rent's rule," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 6, pp. 639–648, 2000.

[228] D. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 288–298, 2004.

[229] J. Gray, C. Twigg, D. Abramson, and P. Hasler, "Characteristics and programming of floating-gate pFET switches in an FPAA crossbar network," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, May 2005, pp. 468–471.

[230] C. Twigg, J. Gray, and P. Hasler, "Programmable floating gate FPAA switches are not dead weight," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2007, pp. 169–172.

[231] G. Mathur, P. Desnoyers, P. Chukiu, D. Ganesan, and P. Shenoy, "Ultra-low power data storage for sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, no. 4, pp. 33:1–33:34, Nov. 2009.

[232] E. Vittoz, "Origins of weak inversion (or sub-threshold) circuit design," in *Sub-threshold Design for Ultra Low-Power Systems*. Springer, 2006, pp. 7–9.

[233] B. Kelly, B. Rumberg, and D. Graham, "An ultra-low-power analog memory system with an adaptive sampling rate," in *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, Boise, ID, August 2012, pp. 302–305.

[234] A. Wang and A. Chandrakasan, "A 180-mv subthreshold FFT processor using a minimum energy design methodology," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, pp. 310–319, Jan. 2005.

[235] K. Kumagai, T. Yamada, H. Iwaki, H. Nakamura, H. Onishi, Y. Matsubara, K. Imai, and S. Kurosawa, "A new SRAM cell design using 0.35 μm CMOS/SIMOX technology," in *Proceedings of the IEEE International SOI Conference*, 1997, pp. 174–175.

[236] M. O'Halloran and R. Sarpeshkar, "A 10-nW 12-bit accurate analog storage cell with 10-aA leakage," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 11, pp. 1985–1996, Nov. 2004.

[237] D. Graham, "A biologically inspired front end for audio signal processing using programmable analog circuitry," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, GA, 2006.

Appendix A

Background on Sub-Threshold Analog Circuits

A.1 Sub-Threshold MOSFET Operation

All integrated circuits (ICs) in this work have been fabricated in standard CMOS processes. Standard CMOS processes are the same mainline processes that are used for logic ICs, such as microcontrollers. As a result, the developments in this work can be tightly integrated with most ICs without any modifications to the manufacturing process.

Various circuit elements are available in standard CMOS. The primary elements used in this work are metal-oxide-semiconductor field-effect transistors (MOSFETs) and poly-insulator-poly (PIP) capacitors, although we have also used metal-insulator-metal (MIM) capacitors when they are available.

Figure A.1 shows an overview of MOSFETs. Complementary N- and P-type transistors are shown. Subplots (a) and (b) show the dependence of the drain current upon the gate voltage for both NMOS and PMOS transistors. MOSFETs have two primary operating regions: sub-threshold and above-threshold; within each of these regions are two sub-regions: saturation and ohmic. In sub-threshold, the drain current is exponentially related to the gate voltage as seen by the linear portions of the traces in the semi-log subplot (a). In above-threshold, the drain current has a square-law relation to the gate voltage as seen by the linear portion of the traces in the root subplot (b). Although the above-threshold region has been the most common, the sub-threshold region has the advantage of lower current (and therefore lower power consumption). Since our objective is low power consumption, most of the circuits in this work operate in the sub-threshold region.

For an nFET in sub-threshold, the relationship between gate voltage and drain current is [232]

$$I_d = I_0 \frac{W}{L} e^{\frac{\kappa V_g}{U_T}} \left(e^{-\frac{V_s}{U_T}} - e^{-\frac{V_d}{U_T}} \right) \tag{A.1}$$

where I_0 is a process-dependent scaler, W and L are the width and length of the channel underneath the gate, $\kappa = \frac{C_{ox}}{C_{ox}+C_{dep}}$[1] is the capacitive coupling from the gate to the channel,

[1] C_{ox} is the oxide capacitance from the gate to the surface and C_{dep} is the depletion capacitance from the body to the channel

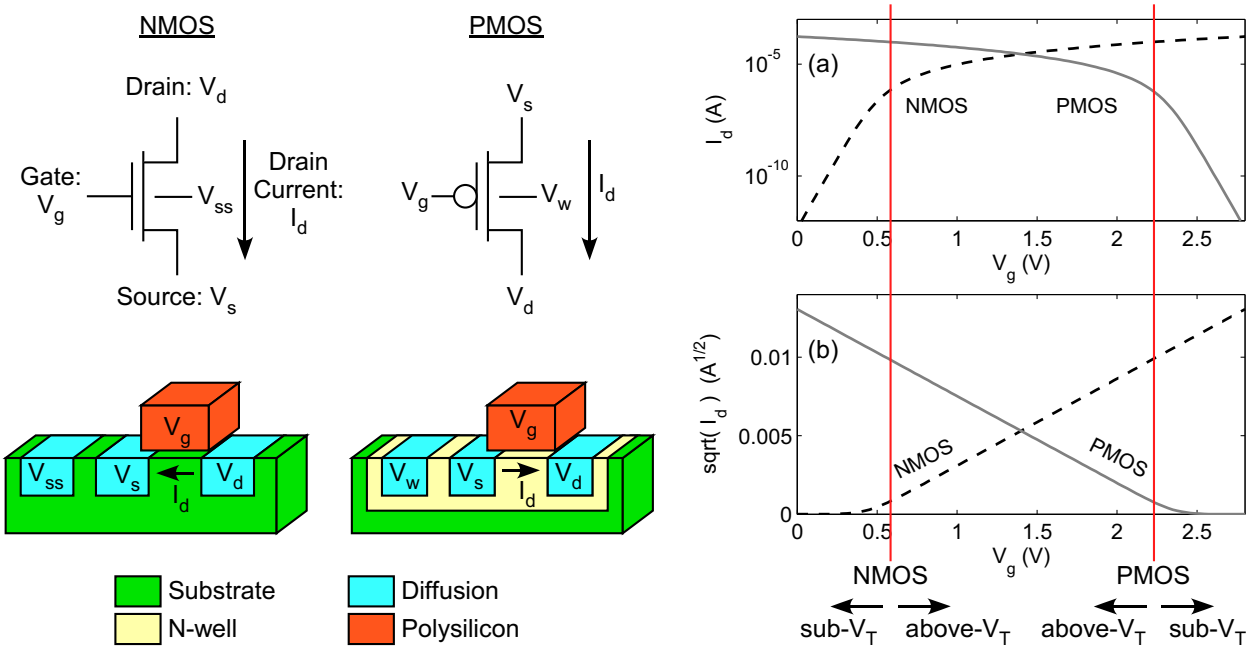

Figure A.1: MOSFET background showing circuit symbols, 3-D renderings, and gate-voltage/drain-current relations.

and $U_T \approx 25.9\text{mV}$ is the thermal voltage. The equation is the same for a pFET, except that all voltages are negative with respect to the transistor's body. For both nFETs and pFETs, when the source is connected to the body (i.e. $e^{-\frac{V_s}{U_T}} = 1$) and the drain voltage is greater than a few tenths of a volt (i.e. $e^{-\frac{V_d}{U_T}} \to 0$), then the expression can be approximated as

$$I_d = I_0 \frac{W}{L} e^{\frac{\kappa V_g}{U_T}} \tag{A.2}$$

This is the saturation sub-region, which is the typical operating point for most transistors in an analog circuit.

A.2 Electronically-Tunable Transconductors

One of the primary circuit blocks used throughout this work, and throughout analog IC design in general, is the operational transconductance amplifier (OTA). An OTA converts a differential voltage input into a current output. This voltage-to-current relationship, or "transconductance," is electronically tunable, which enables the creation of electronically-tunable circuits. The standard symbol for an OTA is shown in Fig. A.2(b), and a basic OTA implementation is shown in Fig. A.2(a). The differential input voltage $V^+ - V^-$ is converted into the current I_{out} according to the following equation for sub-threshold operation

$$I_{out} = I_b \tanh\left(\frac{\kappa(V^+ - V^-)}{2U_T}\right) \tag{A.3}$$

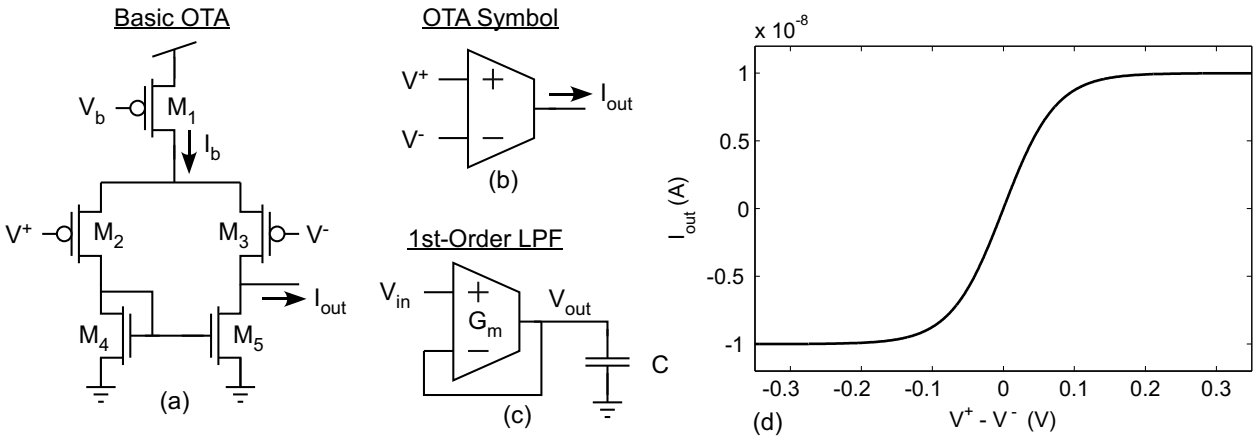

Figure A.2: Overview of operational transconductance amplifiers (OTA). (a) Schematic of the basic OTA. (b) Symbol for an OTA. (c) Using an OTA to create an electronically tunable lowpass filter. (d) OTA input-voltage/output-current relationship.

where I_b is the tunable bias current. This voltage-to-current relationship is plotted in Fig. A.2(d). For small differential input voltages, the transconductance relationship is linear and can be approximated by

$$I_{out} = G_m \left(V^+ - V^-\right) \tag{A.4}$$

where $G_m = \frac{\kappa I_b}{2U_T}$ is the tunable transconductance of the OTA. Although the basic OTA in Fig. A.2(a) has a limited linear range, many techniques exist for increasing the linear range, and we use two of those techniques in Chapters 4 and 9.

Since the bias current I_b sets the transconductance, OTAs can be biased by programmable current sources to create electronically-tunable circuits. For example, in Fig. A.2(c) an OTA is combined with a capacitor to create a first-order lowpass filter with a tunable corner frequency. Equation (A.4) is used to analyze OTA circuits. For example, equating the currents at V_{out} in Fig. A.2(c) yields $G_m \left(V_{in} - V_{out}\right) = sCV_{out}$, and solving for V_{out} yields

$$V_{out} = \frac{V_{in}}{sC/G_m + 1} \tag{A.5}$$

Observe that the time constant is C/G_m, and that it can be tuned by adjusting I_b.

Appendix B

Event Detection Time-Lag and Memory Buffers

For some wake-up applications, the time duration between the occurrence of an event and the assertion of that event is crucial. For example, in applications that record events, the onset of—or even the entirety of—the event may pass before a detection has been asserted. For such situations, a memory buffer may be necessary to record the data. In [233], we described such a memory buffer which included an adaptive sampling approach to minimize the number of samples. In this Appendix, we discuss the detection time-lag for our Hibernets processors and we consider the feasibility of extending our previous memory buffer to a digital-storage format.

B.1 Time Lag to Assert Events

Before discussing the time lag of our Hibernets front-end, let us consider the time lag of the digital back-end. Many low-power microcontrollers and analog-to-digital converters (ADC) can transition from sleep-mode to on-mode in a few microseconds. The greatest time lag is attributed to voltage references, which are required for accurate quantization, and which have typical start-up times on the order of 100μs (e.g. the AD318 low-power reference). For most phenomena, this time lag is insignificant in comparison to the front-end's time lag.

The time lag[1] of our Hibernets front-end (Chapter 3) is primarily caused by the subband magnitude detectors, which cause a delay that is inversely proportional to the subband frequency, f. In Chapter 3, we measured a delay of $4/f$. This result was obtained prior to our more thorough analysis of the magnitude detector (Chapter 5), which now enables us to bias the magnitude detector for the fastest transient response at a given ripple level. The magnitude detector's lag time, or "acquire time," is amplitude dependent because of the adaptive-time-constant filter. If a linear filter were used, the magnitude detector would acquire any signal onset within approximately 7.5 cycles[2]. As described in Section 5.5, when

[1]To determine the time lag of the event detector, we assume that the event appears suddenly, as opposed to slowly intensifying. This avoids any ambiguity as to the precise moment when the event "appears."

[2]For this discussion, the user-defined ripple level is assumed to be 1%, and the acquisition time is defined

using the adaptive-time-constant filter, the magnitude detector can acquire its maximum detectable signal level in approximately 1.5 cycles, and acquires its minimum signals with the same speed as a linear filter ($7.5/f$). In summary, the lag time of a subband is amplitude dependent, and ranges from $1.5/f$ to $7.5/f$, which is consistent with our previously measured $4/f$.

Using this information, we can compare the lag time of the front-end to the lag time of the back-end. If the back-end start-up time is 100μs (i.e. turn-on time of a voltage reference), then the front-end will dominate the overall lag time when the signal frequency is below 1.5 cycles / 100μs = 15kHz. Thus the front end dominates the lag time for most audible frequencies and for most sensor network applications.

B.2 Memory Buffering

Some applications must capture the onset of the event or even the conditions prior to the event. For these applications, a memory buffer may be included in the system, as shown in Fig. B.1(a). This buffer may take the form of an array of sample-and-holds (discrete-time, continuous-value) or an ADC and RAM (discrete-time, discrete-value). During normal operation, the memory buffer runs constantly, holding the recent signal in memory. When an event is detected, the sensor node wakes up and reads out the content of the buffer.

Since the buffer is "always on," power consumption is a major concern. To reduce the number of sampling operations that are required, we have begun to work on a method of compressing the signal as it is recorded. Our approach to this problem is based on the knowledge that the Nyquist sampling criterion—that the sampling frequency must exceed twice the highest frequency of the signal—is excessive for most natural signals, wherein the highest frequencies are rarely present. Hence we have developed a memory system that adapts its sampling rate to the signal [233]. This method is shown in Fig. B.1(b). We use a circuit that identifies local maxima and local minima. From these maxima and minima, time/voltage pairs are generated and stored. Thus, the sampling rate is adapted to equal twice the instantaneous signal frequency, and the number of sampling operations is greatly reduced.

Figure B.2 shows the adaptive-sampler operating on a speech waveform. For this 1.68-second speech waveform, 652 samples were recorded, and the highest frequency captured was 3125Hz. If a constant-sampling-rate technique were used, a total of 10556 samples would have been recorded. This is a 16-fold reduction in the number of sampling operations. We are still determining the criteria for reconstructing from local maxima/minima, but initial work using the Bézier curve equation—shown in Fig. B.2(b)—provides an intelligible reconstruction.

In our initial memory system [233], the data are stored in an array of sample-and-holds (discrete-time, continuous-valued memory). The power consumption of the max/min locator is 1.17μW and the power consumption of the 64-element sample-and-hold array is 2.35μW, yielding a total power of 3.52μW. This is an acceptable power level; however, the sample-and-hold array may not be feasible for moderate-to-high levels of precision. Although the system infrastructure is greatly simplified by using sample-and-holds, the storage duration is limited by leakage. Unfortunately, increasing the storage time by the use of larger capacitive

to be the time to reach 99% of the final value.

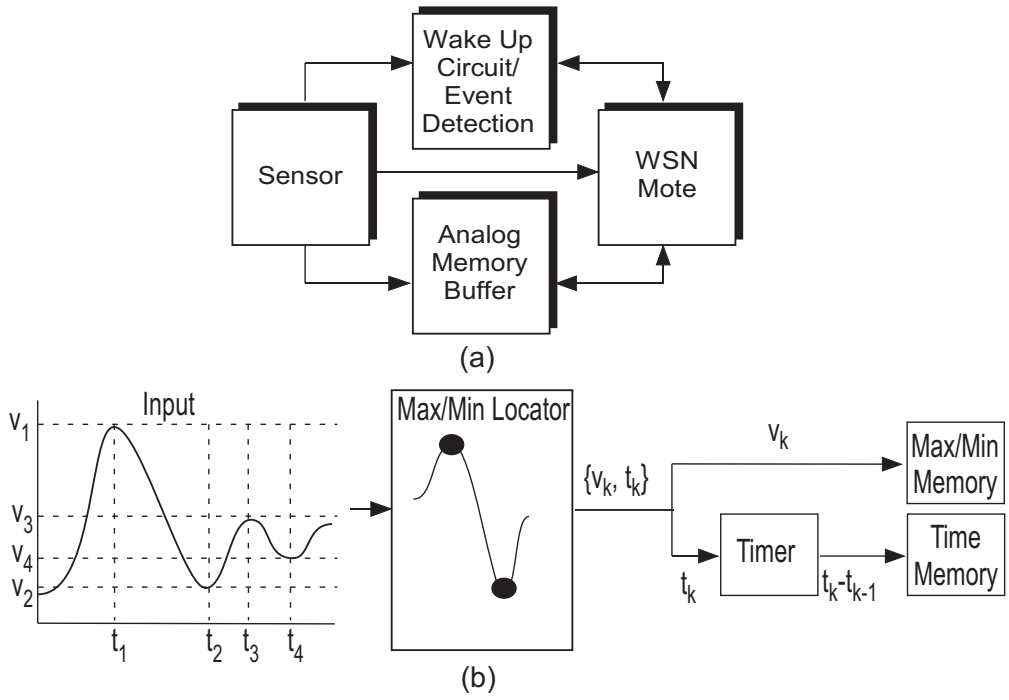

Figure B.1: (a) To avoid the data loss normally associated with sleep states, a low-power memory buffer may be added in parallel with the front-end wake-up detector. This memory system records data from the sensor while the sensor node, or "mote," is allowed to remain in a sleep state, thus providing access to the event onset without sacrificing the energy savings from the sleep state. (b) Our analog memory buffer locates the local maxima and minima in real time and stores their respective amplitudes and times separately.

storage elements or by the use of active leakage reduction comes with a high cost of area or power, respectively. A digital memory buffer may be a better option.

The power of a digital memory buffer system includes an ADC, an impedance buffer to drive the input of the ADC, an SRAM (static random access memory) array, and a state machine to control writing the data to memory. To study the feasibility of a digital memory buffer, we use the power specifications of low-power, off-the-shelf parts. We do not include the power of the state machine that interfaces the ADC with the SRAM: when all parts of the memory system are integrated together, the state machine will consume much less power than the SRAM, because the state machine will have fewer transistors and shorter busses.

The following table summarizes the power consumption of the components.

Component	Part #	Standby	Active Current	Active Time
ADC	AD7467	$I_{s,adc}$=100nA	$I_{a,adc}$=186μA	T=4.7μs
SRAM	23A640	$I_{s,sram}$=200nA	$I_{a,sram}$=6mA	T=4.7μs
Imp. buffer	MCP6041	n/a	I_{buff}=600nA	n/a

The common supply voltage is V_{dd}=1.8V. The total power of the memory system is

$$P_{mem} = V_{dd}\left[I_{buff} + I_{s,adc} + I_{s,sram} + T f_{s,avg}\left(I_{a,adc} + I_{a,sram}\right)\right] \tag{B.1}$$

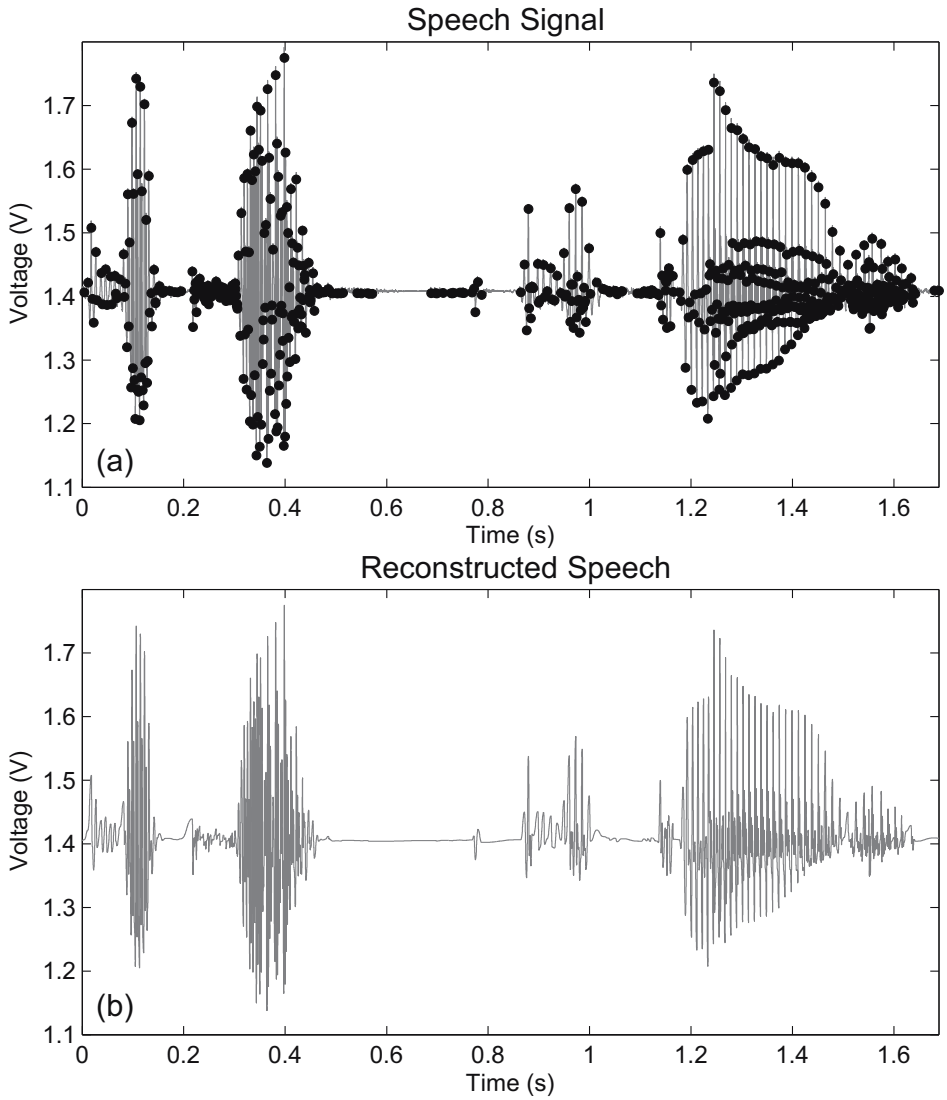

Figure B.2: Adaptive sampling experiment. (a) A speech waveform, shown in gray, is used as the input to the system. The maxima and minima that were detected by the system are shown in black dots. (b) The speech waveform that was reconstructed using the maxima and minima.

where $f_{s,avg}$ is the average sampling frequency. Based on the experiment in Fig. B.2, we will assume $f_{s,avg}$ to be 652-samples / 1.68-seconds. The resulting power is P_{mem}=21.9μW, which is nearly low enough to be acceptable. Approximately half of the power is attributed to the dynamic power of the SRAM array. This dynamic power may be reduced by using a smaller memory array (64kb enables 16.5-seconds of sampling, much more than necessary) and by further increasing the level of compression.

In conclusion, buffering schemes are feasible. The worst-case power consumption, 21.9μW, is less than the power of a sleeping mote. This low power consumption is primarily a result of the adaptive sampler's compression, without which the power consumption would be 326μW

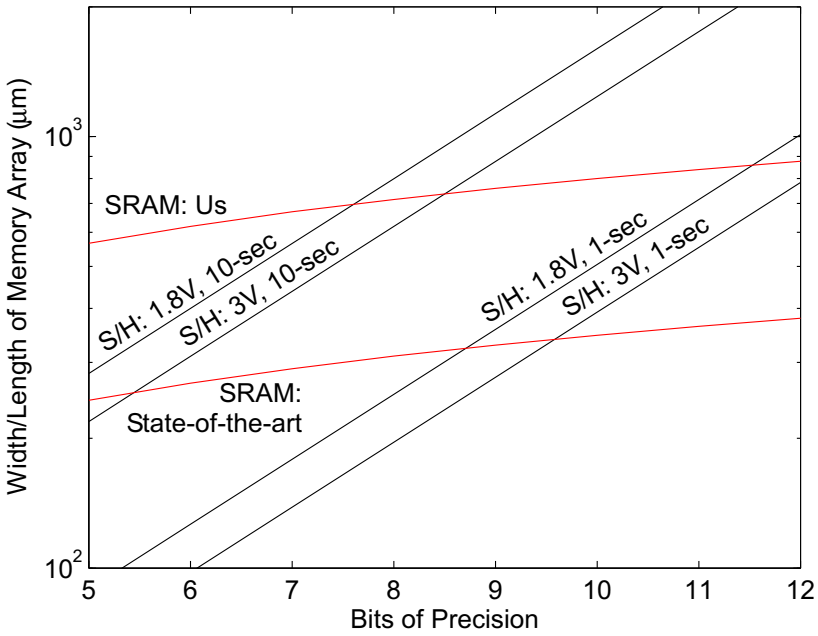

Figure B.3: Size requirements of SRAM memory versus sample-and-hold (S/H) memory. We assume a 0.35μm process. Size is given for a 400-sample memory array. The y-axis shows the width/length (in microns) of a square array.

for the ADC-SRAM implementation, and the number of samples would be too unwieldy to even attempt a sample-and-hold array.

It should be noted that, as a result of the system's adaptive sampling rate, the power consumption and memory requirements will depend upon the signal content. Thus, applications that encounter persistent high-frequency content (either in the signal or background noise) will benefit little from this current implementation of adaptive sampling. Such applications will require processing to obtain a more sparse signal representation prior to adaptive-sampling. Indeed, most applications will benefit from such processing, which will further reduce the number of sampling operations.

Similar to how a digital circuit has an optimal energy-throughput operating point (which is obtained by balancing the standby power with the active power) [234], the memory system's standby power and active power can be balanced by reducing the average sampling frequency to 31Hz (yielding a total power of 3.24μW). The optimal sampling frequency will ultimately depend upon the ADC-memory system, but the idea of a minimal energy sampling frequency should provide designers with a starting point for trading between the energy that is consumed and the lossyness of the compression routine.

In addition to power consumption, area is a major concern for the memory system. The size of an SRAM cell is limited by the pitch of the process. We will assume a 0.35μm process, since that is the process that we are currently using. The cell size in our previously developed SRAM-based switch matrix is $A_{sram,cell}$=160μm^2. This is a very conservative size since the switch cell contains extra transistors and also because analog signal routing was given more than the minimum spacing. In contrast, the cell size of a state-of-the-art 0.35μm SRAM cell

is approximately $A_{sram,cell}$=30μm^2 [235]. The size of an SRAM array scales as follows

$$A_{sram,array} = A_{sram,cell} NB \tag{B.2}$$

where N is the number of samples and B is the wordlength of the samples. The red traces in Fig. B.3 show how the size of an N=400 SRAM array scales with the wordlength.

The size of a sample-and-hold (S/H) cell depends primarily on the sampling capacitor value that is required to maintain the necessary precision. The precision is compromised by thermal noise and by leakage. For storage times greater approximately 10ms, error due to leakage dominates the error due to thermal noise. Assuming that the capacitor area dominates the cell size, the area of the S/H array will be

$$A_{s/h,array} = \frac{N}{\beta} C = \frac{N}{\beta} \frac{I_{leak} T 2^B}{V_{dd}} \tag{B.3}$$

where β=890$\frac{aF}{\mu m^2}$ is the capacitance-per-area, I_{leak}=1fA is the leakage current for a low-leakage *passive* S/H cell [236], T is the hold duration, and V_{dd} is both the supply voltage and the full-scale sampling voltage range. The memory array sizes for SRAM and S/H are compared in Fig. B.3, for different values of V_{dd} and T. For short-duration and low-precision, the S/H implementation appears to offer a smaller array size; however, for this situation the capacitor size is ≈100fF, at which point the assumption that the capacitor dominates the cell size is not valid. So for most realistic applications, SRAM will result in a smaller array implementation, even in a large 0.35μm process.

In summary, sleep mode buffering is feasible, and with an appropriate level of compression should add little to the quiescent power consumption. We have seen that compression is necessary to keep the power consumption low. However, compression may present some obstacles in certain applications. First, the "adaptive sampling rate" form of compression may be inappropriate for some applications. Second, the cost of reconstructing the signal should be low enough that the signal can be reconstructed by the sensor nodes. These obstacles will require further work.

Appendix C

Analysis of the OTA-Based Capacitively-Coupled Current Conveyor

C.1 Derivations for an OTA-based C^4

What follows is a derivation of the equations describing the OTA-based C^4 from Chapter 4. This analysis is based upon the analysis for transistor-based C^4 [237]. In an OTA-based C^4, OTA's replace the source follower and common-source amplifier in the transistor-based C^4. In the following sections, we will derive the transfer function, time constants, Q_{max}, and the relationship between Q_{max} and the transconductances and capacitances.

C.1.1 Transfer Function

First we equate currents at nodes V_a and V_{out}. At node V_a we have

$$sC_1\left(V_a - V_{in}\right) + sC_W V_a + sC_2\left(V_a - V_{out}\right) = G_{m2}\left(V_{out} - V_a\right) \quad \text{(C.1)}$$

$$V_a\left(s\left(C_1 + C_2 + C_W\right) + G_{m2}\right) - sC_1 V_{in} = V_{out}\left(G_{m2} + sC_2\right) \quad \text{(C.2)}$$

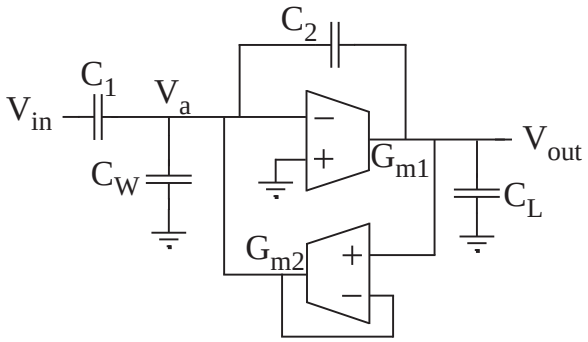

Figure C.1: Schematic of the OTA-C^4 for transfer-function derivation.

and at node V_{out} we have

$$sC_LV_{out} + sC_2(V_{out} - V_a) = -G_{m1}V_a \quad \text{(C.3)}$$

$$V_{out}(sC_L + sC_2) = V_a(sC_2 - G_{m1}) \quad \text{(C.4)}$$

Combining equations C.2 and C.4 we get

$$V_{out}\left(\frac{s\left(C_L + C_2\right)\left(s\left(C_1 + C_2 + C_W\right) + G_{m2}\right)}{sC_2 - G_{m1}} - G_{m2} + sC_2\right) = sC_1V_{in} \quad \text{(C.5)}$$

We define C_T and C_O as

$$C_T = C_1 + C_2 + C_W$$
$$C_O = C_2 + C_L$$

Simplifying equation C.5 yields

$$\frac{V_{out}}{V_{in}} = \frac{s\frac{C_1}{G_{m2}}\left(s\frac{C_2}{G_{m1}} - 1\right)}{s^2\frac{C_OC_T - {C_2}^2}{G_{m1}G_{m2}} + s\left(\frac{C_L}{G_{m1}} + \frac{C_2}{G_{m2}}\right) + 1} \quad \text{(C.6)}$$

The transfer function of a generic bandpass SOS is

$$H(s) = A_v\frac{\frac{\tau}{Q}s}{\tau^2s^2 + \frac{\tau}{Q}s + 1}$$

For the OTA-based C⁴

$$\tau^2 = \frac{C_OC_T - {C_2}^2}{G_{m1}G_{m2}} \quad \text{(C.7)}$$

$$\frac{\tau}{Q} = \frac{C_L}{G_{m1}} + \frac{C_2}{G_{m2}} \quad \text{(C.8)}$$

Solving for Q produces

$$Q = \frac{\tau}{\frac{C_L}{G_{m1}} + \frac{C_2}{G_{m2}}} \quad \text{(C.9)}$$

$$= \frac{\sqrt{C_OC_T - {C_2}^2}}{C_L\sqrt{\frac{G_{m2}}{G_{m1}}} + C_2\sqrt{\frac{G_{m1}}{G_{m2}}}} \quad \text{(C.10)}$$

And the passband gain A_v is

$$numerator = -s\frac{C_1}{G_{m2}}\left(1 - s\frac{C_2}{G_{m2}}\right) \quad \text{(C.11)}$$

$$A_v\frac{\tau}{Q} = -\frac{C_1}{G_{m2}} \quad \text{(C.12)}$$

$$A_v = -\frac{C_1}{G_{m2}}\frac{Q}{\tau} \quad \text{(C.13)}$$

$$= -\frac{C_1}{C_2}\frac{1}{1 + \frac{C_L}{C_2}\frac{G_{m2}}{G_{m1}}} \quad \text{(C.14)}$$

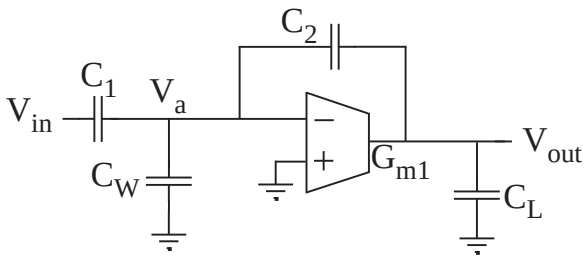

Figure C.2: Schematic of the OTA-C^4 for transfer-function derivation at high frequencies.

C.1.2 C^4 at High Frequencies

Now let's consider the C^4 at high frequencies. At node V_{out}

$$sC_L V_{out} + sC_2 (V_{out} - V_a) = -G_{m1} V_a \quad \text{(C.15)}$$

$$sC_O V_{out} = V_a (sC_2 - G_{m1}) \quad \text{(C.16)}$$

and at node V_a

$$sC_1 (V_a - V_{in}) + sC_W V_a + sC_2 (V_a - V_{out}) = 0 \quad \text{(C.17)}$$

$$V_a C_T = C_1 V_{in} + C_2 V_{out} \quad \text{(C.18)}$$

Combining gives

$$\frac{sC_O V_{out}}{sC_2 - G_{m1}} C_T = C_1 V_{in} + C_2 V_{out} \quad \text{(C.19)}$$

$$V_{out} \left(\frac{sC_O C_T - C_2 (sC_2 - G_{m1})}{sC_2 - G_{m1}} \right) = C_1 V_{in} \quad \text{(C.20)}$$

$$\frac{V_{out}}{V_{in}} = C_1 \frac{sC_2 - G_{m1}}{s \left(C_O C_T - {C_2}^2 \right) + C_2 G_{m1}} \quad \text{(C.21)}$$

$$= -\frac{C_1}{C_2} \frac{1 - s\frac{C_2}{G_{m1}}}{1 + s\frac{C_O C_T - {C_2}^2}{C_2 G_{m1}}} \quad \text{(C.22)}$$

$$= -\frac{C_1}{C_2} \frac{1 - s\tau_f}{1 + s\tau_h} \quad \text{(C.23)}$$

where

$$\tau_f = \frac{C_2}{G_{m1}} \quad \text{(C.24)}$$

$$\tau_h = \frac{C_O C_T - {C_2}^2}{C_2 G_{m1}} \quad \text{(C.25)}$$

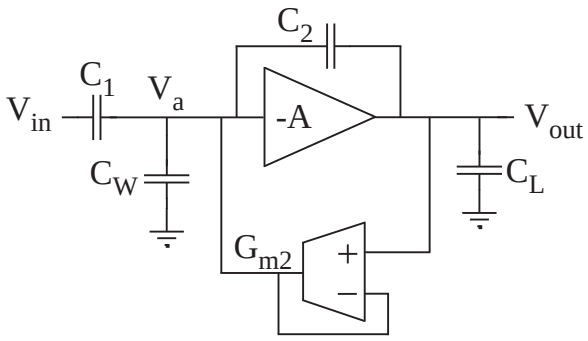

Figure C.3: Schematic for transfer-function derivation at low frequencies

C.1.3 C⁴ at Low Frequencies

Now let's consider the C⁴ at low frequencies. At node V_a

$$sC_1V_{in} + sC_2V_{out} = -G_{m2}V_{out} \tag{C.26}$$

$$V_{out}\left(\frac{sC_2}{G_{m2}} + 1\right) = -s\frac{C_1}{G_{m2}}V_{in} \tag{C.27}$$

$$\frac{V_{out}}{V_{in}} = -\frac{s\frac{C_1}{G_{m2}}}{1 + s\frac{C_2}{G_{m2}}} \tag{C.28}$$

$$= -\frac{C_1}{C_2}\frac{s\frac{C_2}{G_{m2}}}{1 + s\frac{C_2}{G_{m2}}} \tag{C.29}$$

$$= -\frac{C_1}{C_2}\frac{s\tau_l}{1 + s\tau_l} \tag{C.30}$$

where

$$\tau_l = \frac{C_2}{G_{m2}} \tag{C.31}$$

C.1.4 Capacitive Feedthrough

During normal operation we want to keep the corner caused by the capacitive feedthrough time-constant τ_f at a much higher frequency than the low-pass corner caused by τ_h.

$$\tau_f \ll \tau_h \tag{C.32}$$

$$\frac{C_2}{G_{m1}} \ll \frac{C_OC_T - {C_2}^2}{C_2G_{m1}} \tag{C.33}$$

$$2{C_2}^2 \ll C_OC_T \tag{C.34}$$

$$2{C_2}^2 \ll (C_2 + C_L)(C_1 + C_2 + C_W) \tag{C.35}$$

If all capacitors are equal

$$2C^2 \ll 6C^2 \tag{C.36}$$

C.1.5 Solving for Q_{max}

Now let's find the maximum Q value.

$$Q = \frac{\sqrt{C_T C_O - {C_2}^2}}{C_L\sqrt{\frac{G_{m2}}{G_{m1}}} + C_2\sqrt{\frac{G_{m1}}{G_{m2}}}} \frac{\sqrt{\frac{G_{m2}}{G_{m1}}}}{\sqrt{\frac{G_{m2}}{G_{m1}}}} \tag{C.37}$$

$$= \sqrt{\frac{G_{m2}}{G_{m1}}}\sqrt{\frac{C_T C_O - {C_2}^2}{{C_2}^2}} \frac{1}{\frac{C_L}{C_2}\frac{G_{m2}}{G_{m1}} + 1} \tag{C.38}$$

Let

$$b = \frac{G_{m2}}{G_{m1}} \tag{C.39}$$

$$a = \frac{C_L}{C_2} \tag{C.40}$$

$$d = \sqrt{\frac{C_T C_O - {C_2}^2}{{C_2}^2}} \tag{C.41}$$

$$Q = \sqrt{b}d\frac{1}{ab+1} \tag{C.42}$$

$$= \frac{\sqrt{b}d}{1+ab} \tag{C.43}$$

$$= db^{1/2}\left(1+ab\right)^{-1} \tag{C.44}$$

Take the derivative

$$\frac{dQ}{db} = \frac{1}{2}db^{-1/2}\left(1+ab\right)^{-1} + db^{1/2}\left(-a\left(1+ab\right)^{-2}\right) \tag{C.45}$$

$$= \frac{\frac{1}{2}d}{\sqrt{b}}\frac{1-ab}{\left(1+ab\right)^2} \tag{C.46}$$

This equals zero when $1-ab=0$, which is when $b=\frac{1}{a}$ meaning Q_{max} occurs when

$$\frac{C_2}{C_L} = \frac{G_{m2}}{G_{m1}} \tag{C.47}$$

$$Q_{max} = \sqrt{\frac{1}{a}}\frac{d}{1+a\frac{1}{a}} \tag{C.48}$$

$$= \frac{d}{2\sqrt{a}} \tag{C.49}$$

$$= \frac{1}{2}\sqrt{\frac{C_T C_O - {C_2}^2}{C_L C_2}} \tag{C.50}$$

The gain at Q_{max} is

$$A_v = -\frac{C_1}{C_2}\frac{1}{1+a\frac{1}{a}} \tag{C.51}$$

$$= -\frac{C_1}{2C_2} \tag{C.52}$$

$$Q_{max} = \frac{1}{2}\sqrt{\frac{C_T C_O - {C_2}^2}{C_L C_2}} \tag{C.53}$$

$$(2Q_{max})^2 = \frac{C_T C_O - {C_2}^2}{C_L C_2} \tag{C.54}$$

Let $X = (2Q_{max})^2$

$$XC_2C_L = C_T C_O - {C_2}^2 \tag{C.55}$$

$$C_T C_O = C_2\left(XC_L + C_2\right) \tag{C.56}$$

$$= XC_2C_O + {C_2}^2\left(1-X\right) \tag{C.57}$$

$$C_T = XC_2 + \frac{{C_2}^2\left(1-X\right)}{C_O} \tag{C.58}$$

$$= C_2\left(1+\frac{C_L\left(X-1\right)}{C_2+C_L}\right) \tag{C.59}$$

$$C_1 + C_W = C_2\left(\frac{C_L\left(X-1\right)}{C_2+C_L}-\right) \tag{C.60}$$

$$= C_2\left(\frac{C_L\left(X-1\right)}{C_2+C_L}\right) \tag{C.61}$$

$$= \left(X-1\right)\frac{C_2C_L}{C_2+C_L} \tag{C.62}$$

$$= \left(X-1\right)C_2||C_L \tag{C.63}$$

$$= \left(4{Q_{max}}^2-1\right)C_2||C_L \tag{C.64}$$

C.2 OTA-C^4 Noise Analysis

The OTA's are the noise sources in an OTA-C^4. We can determine the noise contribution of each OTA by placing a noise source at the non-inverting terminal of that OTA, grounding the input to the C^4 and calculating the response at the output. Let's start with a noise source at the non-inverting terminal of the feed-forward OTA.

C.2.1 Noise Transfer Function for G_{m1} Noise Source

At node V_{out} we have

$$sC_OV_{out} = G_{m1}v_{1n} + V_a\left(sC_2 - G_{m1}\right) \tag{C.65}$$

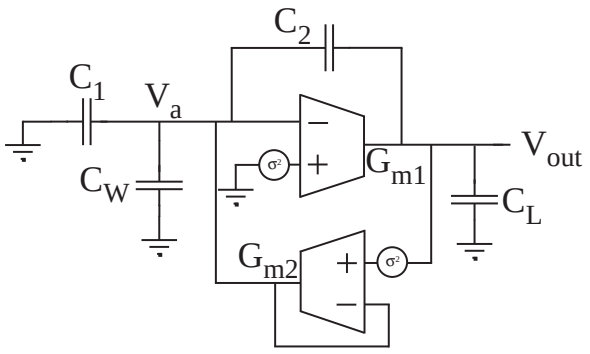

Figure C.4: Schematic of the OTA-C^4 for noise analysis.

and at node V_a we have

$$V_a\left(sC_T + G_{m2}\right) = V_{out}\left(G_{m2} + sC_2\right) \tag{C.66}$$

Combining equations C.65 and C.66 we have

$$V_{out} = v_{1n}\frac{s\frac{C_T}{G_{m2}} + 1}{s^2\frac{C_O C_T - {C_2}^2}{G_{m1}G_{m2}} + s\left(\frac{C_L}{G_{m1}} + \frac{C_2}{G_{m2}+1}\right) + 1} \tag{C.67}$$

C.2.2 Noise Transfer Function for G_{m2} Noise Source

Next we will look at the noise contribution for the feed-back OTA. Starting with node V_{out}

$$sC_O V_{out} = V_a\left(sC_2 - G_{m1}\right) \tag{C.68}$$

and at node V_a

$$V_{out}\left(sC_2 + G_{m2}\right) + G_{m2}i_{2n} = \left(sC_T + G_{m2}\right)V_a \tag{C.69}$$

Combining these equations we get

$$V_{out} = v_{2n}\frac{s\frac{C_2}{G_{m1}} - 1}{s^2\frac{C_T C_O - {C_2}^2}{G_{m1}G_{m2}} + s\left(\frac{C_L}{G_{m1}} + \frac{C_2}{G_{m2}}\right) + 1} \tag{C.70}$$

Since $\frac{C_2}{G_{m1}} = \tau_f$, and τ_f is far outside the passband, we can approximate

$$V_{out} \approx v_{2n}\frac{1}{s^2\frac{C_T C_O - {C_2}^2}{G_{m1}G_{m2}} + s\left(\frac{C_L}{G_{m1}} + \frac{C_2}{G_{m2}}\right) + 1} \tag{C.71}$$

C.2.3 Integrated Noise

Now to combine and integrate the noise. The standard form for the equivalent noise bandwidth of a bandpass filter is

$$\int_0^\infty \frac{\left(\frac{f}{f_0 Q}\right)^2}{\left(1-\left(\frac{f}{f_0}\right)^2\right)^2+\left(\frac{f}{f_0 Q}\right)^2}df = \frac{\pi}{2}\frac{f_0}{Q} \tag{C.72}$$

also

$$\int_0^\infty \frac{1}{\left(1-\left(\frac{f}{f_0}\right)^2\right)^2+\left(\frac{f}{f_0 Q}\right)^2}df = \frac{\pi}{2}f_0 Q \tag{C.73}$$

where

$$\begin{aligned} f_0 &= \frac{1}{2\pi\tau} \\ &= \frac{1}{2\pi}\sqrt{\frac{G_{m1}G_{m2}}{C_O C_T - {C_2}^2}} \qquad \text{(C.74)} \\ Q &= \frac{\sqrt{C_O C_T - {C_2}^2}}{C_L\sqrt{\frac{G_{m2}}{G_{m1}}}+C_2\sqrt{\frac{G_{m1}}{G_{m2}}}} \qquad \text{(C.75)} \end{aligned}$$

The full noise of the circuit is

$$\begin{aligned} \hat{V}_{out}^2 &= \int_0^\infty \frac{{v_{1n}}^2}{\left(1-\left(\frac{f}{f_0}\right)^2\right)^2+\left(\frac{f}{f_0 Q}\right)^2}df \\ &+ \int_0^\infty \frac{{v_{1n}}^2{A_{v1}}^2\left(\frac{f}{f_0 Q}\right)^2}{\left(1-\left(\frac{f}{f_0}\right)^2\right)^2+\left(\frac{f}{f_0 Q}\right)^2}df \\ &+ \int_0^\infty \frac{{v_{2n}}^2}{\left(1-\left(\frac{f}{f_0}\right)^2\right)^2+\left(\frac{f}{f_0 Q}\right)^2}df \\ &+ \int_0^\infty \frac{{v_{2n}}^2{A_{v2}}^2\left(\frac{f}{f_0 Q}\right)^2}{\left(1-\left(\frac{f}{f_0}\right)^2\right)^2+\left(\frac{f}{f_0 Q}\right)^2}df \end{aligned} \tag{C.76}$$

where

$$A_{v1}\frac{\tau}{Q} = \frac{C_T}{G_{m2}} \tag{C.77}$$

$$A_{v1} = \frac{C_T}{G_{m2}}\frac{Q}{\tau} \tag{C.78}$$

$$= \frac{C_T}{G_{m2}}\frac{G_{m1}G_{m2}}{G_{m2}C_L + G_{m1}C_2} \tag{C.79}$$

$$= \frac{G_{m1}C_T}{G_{m2}C_L + G_{m1}C_2} \tag{C.80}$$

$$A_{v2}\frac{\tau}{Q} = \frac{C_2}{G_{m1}} \tag{C.81}$$

$$A_{v2} = \frac{C_2}{G_{m1}}\frac{Q}{\tau} \tag{C.82}$$

$$= \frac{C_2}{G_{m1}}\frac{G_{m1}G_{m2}}{G_{m2}C_L + G_{m1}C_2} \tag{C.83}$$

$$= \frac{G_{m2}C_2}{G_{m2}C_L + G_{m1}C_2} \tag{C.84}$$

$$f_0 Q = \frac{1}{2\pi}\sqrt{\frac{G_{m1}G_{m2}}{C_O C_T - {C_2}^2}}\frac{\sqrt{C_O C_T - {C_2}^2}}{C_L\sqrt{\frac{G_{m2}}{G_{m1}}} + C_2\sqrt{\frac{G_{m1}}{G_{m2}}}} \tag{C.85}$$

$$= \frac{1}{2\pi}\frac{G_{m1}G_{m2}}{C_L G_{m2} + C_2 G_{m1}} \tag{C.86}$$

$$\frac{f_0}{Q} = \frac{1}{2\pi}\sqrt{\frac{G_{m1}G_{m2}}{C_O C_T - {C_2}^2}}\frac{C_L\sqrt{\frac{G_{m2}}{G_{m1}}} + C_2\sqrt{\frac{G_{m1}}{G_{m2}}}}{\sqrt{C_O C_T - {C_2}^2}} \tag{C.87}$$

$$= \frac{1}{2\pi}\frac{C_L G_{m2} + C_2 G_{m1}}{C_O C_T - {C_2}^2} \tag{C.88}$$

Using the integral results listed above

$$\hat{V}_{out}^2 = v_{1n}^2\frac{\pi}{2}f_0 Q + v_{1n}^2 A_{v1}^2\frac{\pi}{2}\frac{f_0}{Q} + v_{2n}^2\frac{\pi}{2}f_0 Q + v_{2n}^2 A_{v2}^2\frac{\pi}{2}\frac{f_0}{Q} \tag{C.89}$$

$$\begin{aligned} = {} & v_{1n}^2\frac{\pi}{2}\frac{1}{2\pi}\frac{G_{m1}G_{m2}}{C_L G_{m2} + C_2 G_{m1}} \\ & + v_{1n}^2\frac{\pi}{2}\left(\frac{G_{m1}C_T}{G_{m2}C_L + G_{m1}C_2}\right)^2\frac{1}{2\pi}\frac{C_L G_{m2} + C_2 G_{m1}}{C_O C_T - C_2^2} \\ & + v_{2n}^2\frac{\pi}{2}\frac{1}{2\pi}\frac{G_{m1}G_{m2}}{C_L G_{m2} + C_2 G_{m1}} \\ & + v_{2n}^2\frac{\pi}{2}\left(\frac{G_{m2}C_2}{G_{m2}C_L + G_{m1}C_2}\right)^2\frac{1}{2\pi}\frac{C_L G_{m2} + C_2 G_{m1}}{C_O C_T - C_2^2} \end{aligned} \tag{C.90}$$

$$\begin{aligned}\hat{V}_{out}^2 &= \frac{v_{1n}^2}{4}\frac{G_{m1}G_{m2}}{C_LG_{m2}+C_2G_{m1}} + \frac{v_{1n}^2}{4}\frac{C_T^2}{C_OC_T-C_2^2}\frac{G_{m1}^2}{G_{m2}C_L+G_{m1}C_2} \\ &+\frac{v_{2n}^2}{4}\frac{G_{m1}G_{m2}}{C_LG_{m2}+C_2G_{m1}} + \frac{v_{2n}^2}{4}\frac{C_2^2}{C_OC_T-C_2^2}\frac{G_{m2}^2}{G_{m2}C_L+G_{m1}C_2}\end{aligned} \tag{C.91}$$

When $Q = Q_{max}$, $G_{m1} = G_{m2}\frac{C_L}{C_2}$ and $G_{m2} = G_{m1}\frac{C_2}{C_L}$. This simplifies the above expression to

$$\hat{V}_{out}^2 = \frac{v_{1n}^2}{4}\frac{G_{m1}}{2C_L} + \frac{v_{1n}^2}{4}\frac{C_T^2}{C_OC_T-C_2^2}\frac{G_{m1}}{2C_2} + \frac{v_{2n}^2}{4}\frac{G_{m2}}{2C_2} + \frac{v_{2n}^2}{4}\frac{C_2^2}{C_OC_T-C_2^2}\frac{G_{m2}}{2C_L} \tag{C.92}$$

Assuming that $C_T^2 \gg C_2^2$, this can be approximated as

$$\hat{V}_{out}^2 = \frac{v_{1n}^2}{4}\frac{G_{m1}}{2C_L} + \frac{v_{1n}^2}{4}\frac{C_TG_{m1}}{2C_2C_O} + \frac{v_{2n}^2}{4}\frac{G_{m2}}{2C_2} + \frac{v_{2n}^2}{4}\frac{G_{m2}C_2^2}{2C_TC_L^2} \tag{C.93}$$

The last term makes contributes very little in relation to the rest of the terms so it can be neglected.

For an OTA the noise is

$$v_n^2 = \frac{NqV_L^2}{I_b} = \frac{NqI_b}{G_m^2} = \frac{NqV_L}{G_m} \tag{C.94}$$

where N is the number of noise sources. We will assume that both OTA's have the same number of noise sources and linear range V_L.

$$\hat{V}_{out}^2 = \frac{NqV_L}{8C_L} + \frac{NqV_LC_T}{8C_2C_O} + \frac{NqV_L}{8C_2} \tag{C.95}$$

Under normal operation, C_L is very large so the first term can be neglected.

$$\hat{V}_{out}^2 = \frac{NqV_L}{8C_2}\left(\frac{C_T}{C_O}+1\right) \tag{C.96}$$

$$Q_{max}^2 = \frac{1}{4}\frac{C_TC_O-C_2^2}{C_LC_2} \approx \frac{1}{4}\frac{C_TC_O}{C_LC_2} \approx \frac{1}{4}\frac{C_T}{C_2} \tag{C.97}$$

There are three cases to consider.

Case 1: $\frac{C_T}{C_O} \gg 1$

$$\hat{V}_{out}^2 \approx \frac{NqV_LC_T}{8C_2C_O} \approx \frac{NqV_LQ_{max}^2}{2C_O} \tag{C.98}$$

$$SNR = 10log_{10}\left(\frac{\left(\frac{V_L}{\sqrt{2}}\right)^2}{\hat{V}_{out}^2}\right) = 10log_{10}\left(\frac{V_LC_O}{NqQ_{max}^2}\right) \tag{C.99}$$

Case 2: $\frac{C_T}{C_O} = 1$

$$\hat{V}_{out}^2 \approx \frac{NqV_L}{4C_2} \tag{C.100}$$

$$SNR = 10log_{10}\left(\frac{2V_LC_2}{Nq}\right) \tag{C.101}$$

Case 3: $\frac{C_T}{C_O} \ll 1$

$$\hat{V}_{out}^2 \approx \frac{NqV_L}{8C_2} \approx \frac{NqV_LQ_{max}^2}{2C_T} \tag{C.102}$$

$$SNR = 10log_{10}\left(\frac{V_LC_T}{NqQ_{max}^2}\right) \tag{C.103}$$

Appendix D

Analysis of the Peak Detector

In Chapter 5, which described our magnitude detector circuit, we described the analysis of the circuit in broad terms. In this Appendix, we provide the full analysis of the peak detector.

The steady-state operating parameters of the peak detector are the tracking level (A_t) and the ripple (R_O). To bias the circuit, the tunable parameters (i.e. the transconductances $G_{m,A}$ and $G_{m,D}$) are set such that the desired operating characteristics are achieved. Thus, biasing requires knowledge of how the operating parameters (A_t and R_O) depend upon the tunable parameters ($G_{m,A}$ and $G_{m,D}$). As such, we have analyzed the peak detector to obtain these relations.

D.1 Problem Setup

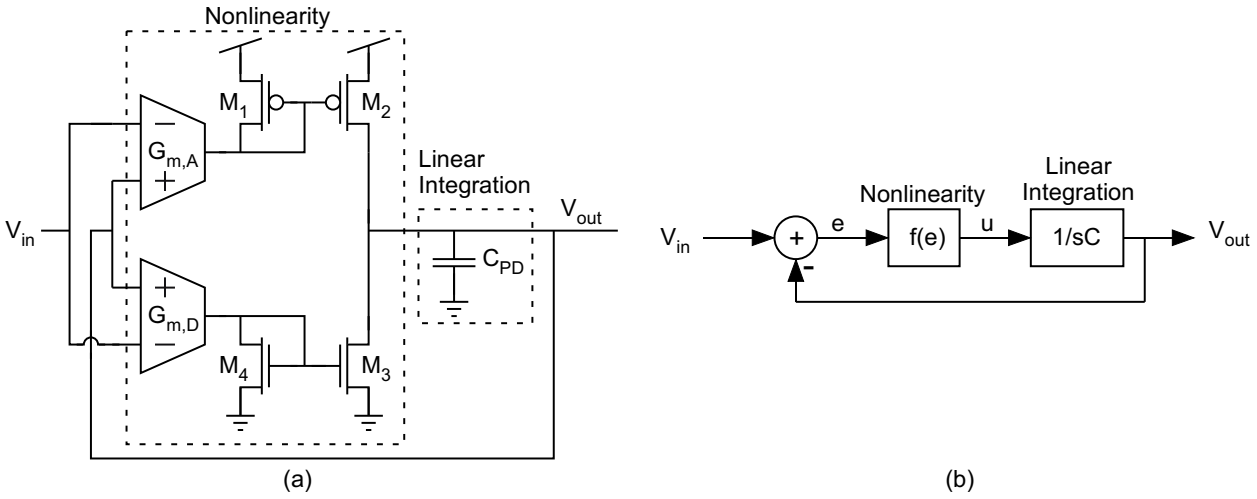

Figure D.1: (a) Peak detector circuit. (b) First-order nonlinear model of the peak detector circuit.

The peak detector has a lowpass form and can be modeled by the system in Fig. D.1,

where the nonlinearity is the piecewise-linear asymmetry formed by the OTAs and current mirrors. The nonlinearity can be written in terms of the transconductances

$$f(e) = \begin{cases} G_{m,a}e, & e > 0 \\ G_{m,d}e, & else \end{cases} \tag{D.1}$$

As an aside, note that (D.1) can be written as

$$f(e) = \frac{G_{m,a} + G_{m,d}}{2}e + \frac{G_{m,a} - G_{m,d}}{2}|e| \tag{D.2}$$

The rectifying nature of the system can be observed from the second term.

The harmonic balance method will be used to analyze the system's steady-state response to a sinusoidal input. Using this method, we will obtain approximate analytical expressions for the dc (i.e. tracking level) and fundamental (i.e. ripple) frequency components. These expressions are approximate because the harmonics created by the nonlinearity are ignored. However, this approximation is justifiable because the circuit has lowpass characteristics and is biased such that the harmonics (as well as the fundamental) are heavily suppressed. Furthermore, our experimental results in Fig. 5.4 show excellent agreement with the derived expressions.

D.1.1 Input/Output Definitions

The input to the system is a sine wave

$$V_{in} = V_{in,pk}\sin(\omega t) \tag{D.3}$$

Within the intended application of spectral analysis, the peak detector will receive its input from narrow-band filters. Thus, the peak detector input will be sine-like, and so performing the analysis only for sine waves is sufficient.

For the harmonic balance method, we specify the form of the output based on the number of frequency components that are expected. Since the peak detector has a lowpass form, the harmonics that are created by the nonlinearity can be ignored. Thus, we write the output as

$$V_{out} = V_{out,pk}\sin(\omega t + \phi) + V_{out,dc} \tag{D.4}$$

$V_{out,dc}$ is the magnitude estimate, which should change in proportion to $V_{in,pk}$. The sine term is the ripple, which is suppressed by the second stage of the magnitude detector.

By defining the circuit's operating parameters as the tracking level

$$A_t = \frac{V_{out,dc}}{V_{in,pk}} \tag{D.5}$$

and the ripple proportion

$$R_O = \frac{V_{out,pk}}{V_{in,pk}} \tag{D.6}$$

we can rewrite the output in terms of those operating parameters

$$V_{out} = V_{in,pk}\left[R_O\sin(\omega t + \phi) + A_t\right] \tag{D.7}$$

A graphical illustration of these parameters is shown in Fig. D.2.

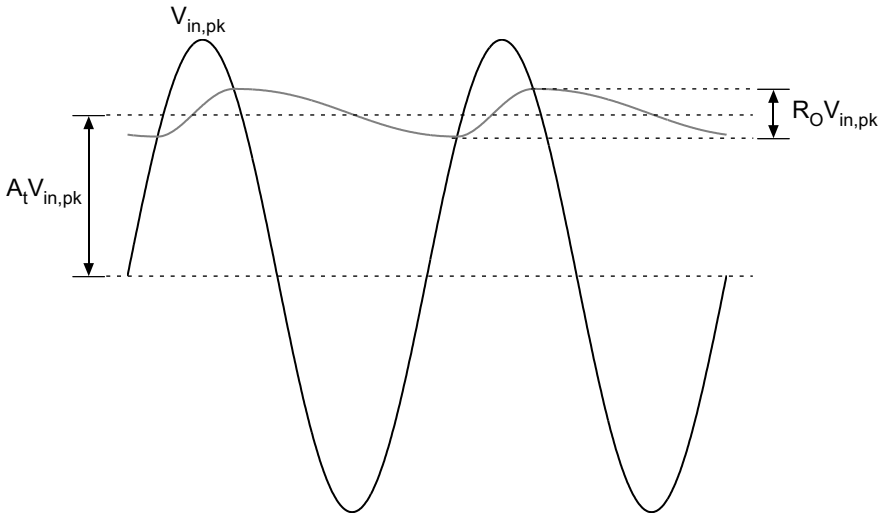

Figure D.2: Illustration of the input/output definitions.

D.2 Solving the Loop

D.2.1 Node e

We begin prior to the nonlinearity, at node e. Combining (D.3) and (D.7), the signal at node e can be written

$$e = V_{in} - V_{out} = V_{in,pk}\left[\sin(\omega t) - R_O \sin(\omega t + \phi) - A_t\right] \tag{D.8}$$

The sinusoids can be combined using the trig identity

$$a\sin x + b\sin(x+\alpha) = \sqrt{a^2 + b^2 + 2ab\cos\alpha}\ \sin\left(x + \gamma\right) \tag{D.9}$$

where

$$\gamma = tan^{-1}\left(\frac{b\sin\alpha}{a + b\cos\alpha}\right) + \begin{cases} 0, & a > 0 \\ \pi, & else \end{cases} \tag{D.10}$$

Using (D.9), (D.8) simplifies to

$$e = V_{in,pk}\left[R_e \sin(\omega t + \gamma) - A_t\right] \tag{D.11}$$

where

$$R_e = \sqrt{1 + R_O^2 - R_O\cos\phi} \tag{D.12}$$

To further simplify, we will define $\theta = \omega t + \gamma$ and write (D.11) as

$$e = V_{in,pk}\left[R_e \sin\theta - A_t\right] \tag{D.13}$$

D.2.2 Node u

The signal at node u can be obtained by inserting (D.13) into (D.1)

$$u = f(e) = \begin{cases} G_{m,a}V_{in,pk}\left[R_e \sin\theta - A_t\right], & \theta > \sin^{-1}\left(\frac{A_t}{R_e}\right) \\ G_{m,d}V_{in,pk}\left[R_e \sin\theta - A_t\right], & else \end{cases} \tag{D.14}$$

Given the form for the output in (D.7), node u can be approximated by its first two Fourier components

$$u = U_0 + U_1 \sin(\theta) \tag{D.15}$$

where

$$U_0 = \frac{1}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} u(\theta)d\theta \tag{D.16}$$

and

$$U_1 = \frac{1}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} u(\theta)\sin\theta d\theta \tag{D.17}$$

The nonlinearity does not present a delay, so a single in-phase component is sufficient.

D.2.2.1 Solving for DC at Node u

To solve for the dc component at node u, insert (D.14) into (D.16) to obtain

$$U_0 = \frac{G_{m,d}V_{in,pk}}{\pi}\int_{-\frac{\pi}{2}}^{\sin^{-1}\left(\frac{A_t}{R_e}\right)}\left[R_e\sin\theta - A_t\right]d\theta + \frac{G_{m,a}V_{in,pk}}{\pi}\int_{\sin^{-1}\left(\frac{A_t}{R_e}\right)}^{\frac{\pi}{2}}\left[R_e\sin\theta - A_t\right]d\theta \tag{D.18}$$

$$= \frac{G_{m,d}V_{in,pk}}{\pi}\left[-R_e\cos\theta - A_t\theta\right]_{-\frac{\pi}{2}}^{\sin^{-1}\left(\frac{A_t}{R_e}\right)} + \frac{G_{m,a}V_{in,pk}}{\pi}\left[-R_e\cos\theta - A_t\theta\right]_{\sin^{-1}\left(\frac{A_t}{R_e}\right)}^{\frac{\pi}{2}} \tag{D.19}$$

Using the identity

$$\cos\left(\sin^{-1}(x)\right) = \sqrt{1-x^2} \tag{D.20}$$

and grouping terms, we obtain

$$U_0 = \frac{(G_{m,A} - G_{m,D})V_{in,pk}}{\pi}\left(R_e\sqrt{1-\frac{A_t^2}{R_e^2}} + A_t\sin^{-1}\left(\frac{A_t}{R_e}\right)\right) - \frac{(G_{m,a}+G_{m,d})V_{in,pk}A_t\pi}{2} \tag{D.21}$$

D.2.2.2 Solving for the Fundamental at Node u

To solve for the fundamental component at node u, insert (D.14) into (D.17) to obtain

$$U_1 = \frac{G_{m,d}V_{in,pk}}{\pi}\int_{-\frac{\pi}{2}}^{\sin^{-1}\left(\frac{A_t}{R_e}\right)}\left[R_e\sin\theta - A_t\right]\sin\theta d\theta + \frac{G_{m,a}V_{in,pk}}{\pi}\int_{\sin^{-1}\left(\frac{A_t}{R_e}\right)}^{\frac{\pi}{2}}\left[R_e\sin\theta - A_t\right]\sin\theta d\theta \tag{D.22}$$

Looking at just the integral, we obtain

$$\int_{\theta_1}^{\theta_2} [R_e \sin\theta - A_t] \sin\theta d\theta = \int_{\theta_1}^{\theta_2} \left[\frac{R_e}{2} - \frac{R_e}{2}\cos(2\theta) - A_t \sin\theta\right] d\theta \quad \text{(D.23)}$$

$$= \left[\frac{R_e}{2}\theta - \frac{R_e}{4}\sin(2\theta) + A_t \cos\theta\right]_{\theta_1}^{\theta_2} \quad \text{(D.24)}$$

Inserting (D.24) into (D.22) and grouping terms yields

$$U_1 = \frac{(G_{m,A} + G_{m,D})R_e V_{in,pk}}{4} - \frac{(G_{m,A} - G_{m,D})V_{in,pk}}{\pi}\left[A_t \cos\left(\sin^{-1}\left(\frac{A_t}{R_e}\right)\right)\right.$$
$$\left. - \frac{R_e}{4}\sin\left(2\sin^{-1}\left(\frac{A_t}{R_e}\right)\right) + \frac{R_e}{2}\sin^{-1}\left(\frac{A_t}{R_e}\right)\right] \quad \text{(D.25)}$$

Using the identity in (D.20) as well as the following identity

$$\sin\left(2\sin^{-1}(x)\right) = 2x\sqrt{1-x^2} \quad \text{(D.26)}$$

we can simplify (D.25) to

$$U_1 = \frac{(G_{m,A} + G_{m,D})R_e V_{in,pk}}{4} - \frac{(G_{m,A} - G_{m,D})V_{in,pk}}{\pi}\left[\frac{A_t}{2}\sqrt{1 - \frac{A_t^2}{R_e^2}} + \frac{R_e}{2}\sin^{-1}\left(\frac{A_t}{R_e}\right)\right] \quad \text{(D.27)}$$

D.2.3 Node V_{out}

The output is related to node u as

$$V_{out} = \frac{u}{sC} \quad \text{(D.28)}$$

Equating the frequency components in (D.7) and (D.15) yields

$$V_{out,dc} = V_{in,pk}A_t = \frac{U_0}{|sC|} \quad \text{(D.29)}$$

at dc and

$$V_{out,pk} = V_{in,pk}R_O = \frac{U_1}{|sC|} \quad \text{(D.30)}$$

at the fundamental frequency.

D.3 Balancing the Terms

We can now obtain the analytical expressions that relate the operating parameters A_t and R_O to the tuning parameters $G_{m,A}$ and $G_{m,D}$.

D.3.1 Tracking Level

To solve the loop at dc, we write (D.29) as

$$U_0 = |sC|V_{out,dc} \tag{D.31}$$

Since this is dc, s=0 and therefore U_0=0. Setting (D.27) to zero

$$0 = (G_{m,A} - G_{m,D})\frac{V_{in,pk}}{\pi}\left(R_e\sqrt{1-\frac{A_t^2}{R_e^2}} + A_t \sin^{-1}\left(\frac{A_t}{R_e}\right)\right) - (G_{m,A} + G_{m,D})\frac{V_{in,pk}A_t}{2} \tag{D.32}$$

and dividing through by $\frac{V_{in,pk}A_t}{2}$

$$0 = (G_{m,A} - G_{m,D})\frac{2}{\pi}\left(\frac{R_e}{A_t}\sqrt{1-\frac{A_t^2}{R_e^2}} + \sin^{-1}\left(\frac{A_t}{R_e}\right)\right) - (G_{m,A} + G_{m,D}) \tag{D.33}$$

For the moment, let us define

$$L = \frac{2}{\pi}\left(\frac{R_e}{A_t}\sqrt{1-\frac{A_t^2}{R_e^2}} + \sin^{-1}\left(\frac{A_t}{R_e}\right)\right) \tag{D.34}$$

Now a few manipulations

$$\begin{aligned} 0 &= (G_{m,A} - G_{m,D})L - (G_{m,A} + G_{m,D}) && \text{(D.35)}\\ 0 &= G_{m,A}(L-1) - G_{m,D}(L+1) && \text{(D.36)}\\ G_{m,A}(L-1) &= G_{m,D}(L+1) && \text{(D.37)}\\ \frac{G_{m,A}}{G_{m,D}} &= -\frac{1+L}{1-L} = \frac{L+1}{L-1} && \text{(D.38)} \end{aligned}$$

Take the log of both sides and multiply by $\frac{1}{2}$

$$\frac{1}{2}\ln\left(\frac{G_{m,A}}{G_{m,D}}\right) = \frac{1}{2}\ln\left(\frac{L+1}{L-1}\right) \tag{D.39}$$

and use the hyperbolic function identity

$$\coth^{-1}(x) = \frac{1}{2}\ln\left(\frac{x+1}{x-1}\right) \tag{D.40}$$

to obtain

$$\frac{1}{2}\ln\left(\frac{G_{m,A}}{G_{m,D}}\right) = \coth^{-1}(L) \tag{D.41}$$

Now reinsert (D.34) to obtain the final equation

$$\frac{1}{2}\log\left(\frac{G_{m,A}}{G_{m,D}}\right) = \coth^{-1}\left(\frac{2}{\pi}\left(\frac{R_e}{A_t}\sqrt{1-\frac{A_t^2}{R_e^2}} + \sin^{-1}\left(\frac{A_t}{R_e}\right)\right)\right) \tag{D.42}$$

D.4 Ripple

From (D.30), we can relate U_1 to the ripple as

$$R_O = \frac{V_{out,pk}}{V_{in,pk}} = \frac{U_1}{|sC|}\frac{1}{V_{in,pk}} \tag{D.43}$$

$$\omega C R_O = \frac{U_1}{V_{in,pk}} \tag{D.44}$$

Inserting U_1 yields

$$\omega C R_O = \frac{(G_{m,A}+G_{m,D})R_e}{4} - \frac{G_{m,A}-G_{m,D}}{\pi}\left[\frac{A_t}{2}\sqrt{1-\frac{A_t^2}{R_e^2}} + \frac{R_e}{2}\sin^{-1}\left(\frac{A_t}{R_e}\right)\right] \tag{D.45}$$

Defining $R_g = \frac{G_{m,A}}{G_{m,D}}$ and dividing through by $G_{m,D}$

$$\frac{R_O\omega C}{G_{m,D}} = \frac{(R_g+1)R_e}{4} - \frac{R_g-1}{\pi}\left[\frac{A_t}{2}\sqrt{1-\frac{A_t^2}{R_e^2}} + \frac{R_e}{2}\sin^{-1}\left(\frac{A_t}{R_e}\right)\right] \tag{D.46}$$

Dividing by R_e

$$\frac{R_O\omega C}{R_e G_{m,D}} = \frac{R_g+1}{4} + \frac{1-R_g}{\pi}\left[\frac{A_t}{R_e 2}\sqrt{1-\frac{A_t^2}{R_e^2}} + \frac{1}{2}\sin^{-1}\left(\frac{A_t}{R_e}\right)\right] \tag{D.47}$$

$$\frac{\omega C_{PD} R_O}{G_{m,D} R_e} = \frac{R_g+1}{2} + \frac{1-R_g}{\pi}\sin^{-1}\left(\frac{A_t}{R_e}\right) + \frac{1-R_g}{\pi}\frac{A_t}{R_e}\sqrt{1-\frac{A_t^2}{R_e^2}} \tag{D.48}$$

D.5 Conclusions

We have derived two equations—(D.42) and (D.48)—which enable us to purposefully bias the peak detector circuit. The biasing procedure is described in Section 5.3.3. These equations describe the steady-state characteristics of an asymmetric integrator in response to a sinusoidal input. Asymmetric integration is relevant in other areas, such as in neuronal dynamics, and so this analysis may be of broader interest than for peak detectors.